NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

ANALYZING ANTI-TERRORIST TACTICAL EFFECTIVENESS OF PICKET BOATS FOR FORCE PROTECTION OF NAVY SHIPS USING X3D GRAPHICS AND AGENT-BASED SIMULATION

by

James William Harney

March 2003

Thesis Advisor:	Donald P. Brutzman
Thesis Co-advisor:	Curtis L. Blais
Thesis Co-advisor:	Gordon Schacher
Thesis Co-advisor:	John Hiles

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** March 2003	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: **Analyzing Anti-Terrorist Tactical Effectiveness of Picket Boats for Force Protection of Navy Ships Using X3D Graphics and Agent-Based Simulation**			**5. FUNDING NUMBERS**
6. AUTHOR **James William Harney**			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) Space and Naval Warfare Systems Center, San Diego California (SSC San Diego), Defense Modeling and Simulation Office (DMSO)			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			**12b. DISTRIBUTION CODE** A

13. ABSTRACT

Despite the many advances achieved within both Modeling and Simulation and Information Technology over the past several decades, practical application of such technology remains under-utilized by operational units in the United States Navy. Furthermore, when such technology has been deployed in the last decade it has been to exercise operator proficiency or increase C4I battlespace awareness. Few tools have allowed operational warfighters to run 'what-if' simulation scenarios to aid in development of tactical plans for executing published doctrine.

The approach taken in this thesis is to select an exemplar warfare area, in this case Anti-Terrorism and Force Protection for Navy ships, and through research and development to identify, develop, and deploy the necessary modeling and simulation (M & S) technologies to demonstrate a prototypical planning tool that can be used by today's deployed warfighter. All research and work is conducted in a web-based, 'user-centric' fashion utilizing a combination of user-driven and agent-based control of entities for simulation iterations, along with various open source technologies which include Extensible 3D Graphics (X3D), Scalable Vector Graphics (SVG), and Extensible Markup Language (XML). Conventions are demonstrated for the integration of the many academic disciplines utilized during this research to achieve automatic generation of tactically significant scenarios. In order to give the end-user the greatest insight towards potential drawbacks in the tactical planning against surface-borne terrorist threats, various 2D and 3D media provide both real-time and non-real time scenario playback.

The result of this work is a fully integrated, prototypical, Java-based application that demonstrates how various Open-Source, web-based technologies can be applied in order to provide the tactical operator with tools to aid in Force Protection planning. Scenarios can be auto generated, viewed, analyzed, and manipulated by end users with little to no computer experience necessary beyond requirements for operation of a desktop personal computer (PC) in the Information Technology for the 21st Century (IT-21) environment at sea. This approach has broad applicability to improve the tactical awareness and defensive posture of ships defending against terrorist attacks in port.

14. SUBJECT TERMS Virtual Environments, Extensible 3D Graphics, X3D, Scalable Vector Graphics, SVG, Force Protection, Anti-Terrorism, Extensible Markup Language, XML, Java, Scenario Generation, DIS-Java-VRML, Extensible Modeling and Simulation Framework (XMSF), SAVAGE, Distributed Interactive Simulation, NPSNET-V			**15. NUMBER OF PAGES** 252
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

ANALYZING ANTI-TERRORIST TACTICAL EFFECTIVENESS OF PICKET BOATS FOR FORCE PROTECTION OF NAVY SHIPS USING X3D GRAPHICS AND AGENT-BASED SIMULATION

James W. Harney
Lieutenant, United States Navy
B.S., United States Naval Academy, 1996

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2003

Author:	James William Harney
Approved by:	Donald P. Brutzman, Thesis Advisor
Approved by:	Curtis L. Blais, Thesis Co-advisor
Approved by:	Gordon Schacher, Thesis Co-advisor
Approved by:	John Hiles, Thesis Co-advisor
Approved by:	Peter Denning, Chairman, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Despite the many advances achieved within both Modeling and Simulation and Information Technology over the past several decades, practical application of such technology remains under-utilized by operational units in the United States Navy. Furthermore, when such technology has been deployed in the last decade it has been to exercise operator proficiency or increase C4I battlespace awareness. Few tools have allowed operational warfighters to run 'what-if' simulation scenarios to aid in development of tactical plans for executing published doctrine.

The approach taken in this thesis is to select an exemplar warfare area, in this case Anti-Terrorism and Force Protection for Navy ships, and through research and development to identify, develop, and deploy the necessary modeling and simulation (M & S) technologies to demonstrate a prototypical planning tool that can be used by today's deployed warfighter. All research and work is conducted in a web-based, 'user-centric' fashion utilizing a combination of user-driven and agent-based control of entities for simulation iterations, along with various open source technologies which include Extensible 3D Graphics (X3D), Scalable Vector Graphics (SVG), and Extensible Markup Language (XML). Conventions are demonstrated for the integration of the many academic disciplines utilized during this research to achieve automatic generation of tactically significant scenarios. In order to give the end-user the greatest insight towards potential drawbacks in the tactical planning against surface-borne terrorist threats, various 2D and 3D media provide both real-time and non-real time scenario playback.

The result of this work is a fully integrated, prototypical, Java-based application that demonstrates how various Open-Source, web-based technologies can be applied in order to provide the tactical operator with tools to aid in Force Protection planning. Scenarios can be auto generated, viewed, analyzed, and manipulated by end users with little to no computer experience necessary beyond requirements for operation of a desktop personal computer (PC) in the Information Technology for the 21st Century (IT-21) environment at sea. This approach has broad applicability to improve the tactical awareness and defensive posture of ships defending against terrorist attacks in port.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENTS

The author would like to acknowledge the direct financial support of the Space and Naval Warfare Systems Center, San Diego California (SSC San Diego), and the Defense Modeling and Simulation Office (DMSO), for sponsoring portions of this research. The author would also like to thank the following list of people whose assistance, support, and contributions made this thesis possible:

- Don Brutzman, Curt Blais, Gordon Schacher, and John Hiles from the Naval Postgraduate School (NPS) for their guidance and support during the course of this thesis research.
- Alan Hudson, Justin Couch, and Stephen Matsuba from Yumetech, Inc. for both their work and contributions on the Xj3D Toolkit for X3D, but also for providing copious amounts of information and guidance for development of this thesis.
- Andrezj Kapolka from NPS for his contributions and assistance with the application architecture design and integration with NPSNET-V and Xj3D.
- Don MacGregor and Ekrem Serin from NPS for their guidance on Java network programming and integration with this thesis project.
- David Colleen of Planet 9 Studios for providing instruction on advanced 3D content authoring and assistance for the I/ITSEC 2002 conference.
- Nick Polys from Virginia Polytechnic Institute and State University for taking the time to introduce me to advanced Web 3D authoring techniques and theories at the onset of this thesis work.
- George Lawler from NPS for his contributions, assistance, and work in assistance with the development of the initial user-interface.
- Jeff Weekley and Doug Horner from NPS for providing general assistance with learning both 2D and 3D graphics and providing useful feedback during the course of this research.
- My brother Matt and the many others like him who are currently deployed carrying out the fight against terrorism.
- My wife Tanya, and daughter Laurie for their encouragement, tolerance, and perseverance during my late nights on the road to thesis completion.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Si vis pacem para bellum. If you want peace, prepare for war.

--Roman maxim circa 5th Century A.D.

A. PROBLEM STATEMENT

Despite the many advances achieved within the fields of Modeling and Simulation (M & S) and Information Technology (IT) over the past several decades, practical application of such technology remains under-utilized by operational units in the United States Navy. Furthermore, when such technology has been deployed in the last decade it has been to exercise operator proficiency and increase C^4I battlespace awareness, but not for the warfighter to run 'what-if' scenarios that might aid in development of tactical plans for employing established doctrine.

This thesis presents the investigation, research, and development of an exemplar Anti-Terrorist / Force Protection (AT/FP) application using current web-based technologies. This project can aid the warfighter in gaining insights for planning AT/FP doctrine for navy ships, specifically against the water-side surface-borne terrorist threat. Future extensions will do air, subsurface, and land-side also. This application is accomplished by planning doctrinal implementation with 2D graphics, running 'what-if' scenarios in various web-based graphics formats, as well as through the capability for conducting basic statistical analysis to facilitate single unit-level planning. The end result is a prototypical planning tool that can be used for U.S. Navy AT/FP plan editing, visualization, and training against the surface-borne terrorist threat.

B. OVERVIEW

Joint Publication 1-02 defines <u>terrorism</u> as,

> The calculated use of violence or threat of violence to inculcate fear; intended to coerce or intimidate governments or societies in the pursuit of goals that are generally political, religious, or ideological." [JP1 2002]

Over the past decade, United States armed forces have increasingly been the target of terrorist organizations. One publicly stated declaration of this threat was Usama Bin Laden's comments on May 22, 1998:

> Killing Americans and their Allies, civilian and military, is an individual duty for every Muslim…we do not differentiate between those dressed in military uniforms and civilians. [JMOCN 2002]

Specifically, and prior to the Al-Qaida sponsored terrorist attack on September 11th, 2001, Joint Vision 2020 stated that with U.S. conventional forces continuing to maintain a force dominance in the twenty-first century,

> the appeal of asymmetric approaches and the focus on the development of niche capabilities will increase. [JV2020 2000]

With these ideas and the terrorist attacks against the USS Cole (DDG 67) in October 2002 (Figure 1) and the French Oil Tanker, Limburg [GUARDIAN 2002] in the same port in mid-2002 in mind, it is evident that U.S. naval forces will continue to pose attractive targets for terrorist organizations, whether deployed, overseas, or inport within the continental United States (CONUS). As a result, an open-ended research question has been how the U. S. Department of Defense (DoD) can best leverage our IT advantages in order to bolster both our counter-terrorist capabilities and those of allied countries. [JMOCN 2002]

The approach presented in this thesis is specifically aimed at providing advanced analytical and visualization capabilities to everyday defenders directly onboard operational ships, at sea and in ports.


Figure 1. External Damage on USS COLE after Terrorist Attack in Aden Harbor, Yemen October 12, 2000. 17 Individuals were killed in action, 39 were wounded in action. [JMOCN 2002]

C. MOTIVATION

On October 12th, 2000, the USS COLE (DDG 67) was attacked in Aden Harbor, Yemen. Al-Qaida sponsored operatives maneuvered a fiberglass skiff laden with approximately 1/3 ton of TNT equivalent near COLE while she was refueling, and then detonated the explosions onboard the skiff. [CCOI 01] The operatives sacrificed themselves to carry out the attack, which resulted in 17 U.S. sailors being killed and 37 wounded in action from a crew of approximately 320 total sailors. [JMOCN 2002] Although the ship was saved by the crew from being sunk by the attack, COLE required extensive repairs that kept her under repair through April 2002.

Worth considering is a mission technological opportunity: if M&S tools and 3D visualization had been available to the Force Protection planners before entering this port, perhaps a profound improvement in self-defense posture might have prevented the approach? Unfortunately, it is not possible to change what happened in Yemen, but only

hope to have a positive impact on future events. Hopefully, by empowering operators at sea with M&S tools, we can aid in training for future situations in which sailors will be in harm's way and save lives.

D. OBJECTIVES

This thesis demonstrates and evaluates both functionality and effectiveness of employing web-based modeling and simulation technologies such as X3D graphics and agent-based simulation for planning Anti-Terrorist and Force Protection (AT/FP) measures. The objective is to determine how these technologies can specifically be leveraged to give planners greater insight than has been provided by traditional wardroom discussion augmented by paper chart, "back of the envelope" conclusions, and other text-based information. To accomplish this objective, several items must be done:

1) Identification of visualization, data storage, and any other applicable technologies to be utilized must be made.

2) Design and Implementation of an application framework must be designed and implemented in order to allow development of the various models, views, and controls representing different entities such as ships, land, small boats, terrorists, etc within our scenarios.

3) Development of a 'user-centric' interface to the application has to be developed in order to provide tactical operators the information needed in an easy-to-use manner.

4) Demonstration of a use-case presenting the application's utility and how it can be leveraged for education and training in Force Protection.

E. THESIS ORGANIZATION

Chapter II reviews background and related information on defending against the asymmetrical threat. Synopses are provided for the various technologies and techniques investigated, developed, and leveraged during this thesis work. Chapter III explains the problem definition, design, and proposed solution in formal detail. Chapter IV provides an in-depth analysis and demonstration of how a web-based X3D reconstructive scene ie

developed depicting the attack on the USS COLE. It also shows how the components and methodology applied can be extended and used in other problem domains. Chapter V reviews the physics modeling utilized in the real-time model developed. Chapter VI discusses the design process of developing the scenario generation interface in a 'user-centric' fashion following current industry best practices. Chapter VII examines the specific model design for the thesis work. Chapter VIII analyzes the agent-based software design and implementation for autonomous control of role-based entities within the AT/FP application. Chapter IX presents a use case that shows how an end-user can develop a force protection plan and possibly gain insights to potential tactical short-comings. Chapter X shows how the M&S technology developed can be utilized for the training and education of Anti-terrorist measures within the Limited Objective Experiment framework for AT/FP doctrinal development within Commander, U.S. Pacific Fleet(COMPACFLT). Finally, Chapter XI summarizes the conclusions and recommendations for future work of this thesis. The appendices present amplifying information where appropriate as well as information on obtaining all graphics and programming source code produced in conjunction with this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND AND RELATED WORK

A. INTRODUCTION

This chapter briefly reviews the numerous concepts that are the root basis for understanding the multiple technologies leveraged in the conduct of this thesis. Subject matter includes a brief overview of all technologies, design paradigms, and graphics standards employed. Further explanation and study of the topics may be found in the list of references at the conclusion of this thesis.

B. ROLE OF 3D VISUALIZATION IN MODELING ANDSIMULATIONTECHNOLOGY

As shown in [Nicklaus 2001], [Dickie 2002], and in other research domains, visualization technologies have played numerous roles in the recent years. Capabilities range from quick visual validation of the interactions of virtual entities represented by various models to providing spatial awareness for fully elaborated battlespace descriptions such as Marine Corps Operations Orders. [Nicklaus 2001] In the context of this thesis, web-based 3D visualizations are used in a combination of ways to achieve the goal of providing a usable, web-based modeling and simulation tool for defensive planning against small, fast, surface-ship terrorist threats. This is a nascent field: few repeatable examples are widely available. Numerous necessary technologies are needed and synopsized in the next sections.

C. OPEN-SOURCE CONCEPTS AND TECHNOLOGIES

Open-Source software is freely available for any use, including modification and redistribution. The first formal statement of the official Open Source definition appeared in 1997 by Bruce Perens [OSI 2002]. This definition has continued to be refined and maintained by the Open Source Initiative, a non-profit corporation. [OSI 2002] In short, the definition states that besides providing access to source code, the distribution terms of compliant open-source software must conform to the following set of criteria:

- Be a free redistribution,
- Include source code,
- Allow modification and redistribution,
- Maintain integrity of the original author's source so users know who is responsible for maintenance and support,
- Not discriminate against persons or groups,
- Not discriminate against fields of endeavor,
- Distribution of license with the software must apply to others without execution of addition requirements to keep software from being secured away through various legal means,
- The license is not specific to a product, and
- The license should not restrict other software that might not be open source. [OSI 2002]

The general idea behind the Open Source movement in the programming community has been that when software can be freely read, redistributed, and modified, it accelerates the development process as compared to traditionally closed development cycles of major infrastructure such as an operating system, graphics format standard, etc.

The Open Source idea has existed for a number of years, but only in the recent past has it seen exceptionally widespread use in the business and government worlds. Examples include the embracing of the Linux operating system by companies such as IBM, Oracle, and Sun Microsystems [LINUX 2002], as well as the formation of the Extensible 3D (X3D) Graphics Computer Aided Design (CAD) working group by the Intel Corporation. [INTEL 2002] These companies have shown that successful business models can be based upon open standards by focusing on services.

As a result, the decision was made early in the formation of this thesis to rely upon open standards, open architectures, and web-based programming languages as the basis for which to implement application development.

D. EXTENSIBLE MARKUP LANGUAGE (XML)

The Extensible Markup Language (XML) was developed by the World Wide Web Consortium (W3C) as a subset of the Standard Generalized Markup Language (SGML) with the initial goal of being able to define data with the simplicity of Hyper Text Markup Language (HTML). HTML is primarily intended for the display of data to the end-user while retaining the flexibility and extensibility of SGML. [Hunter 2001] HTML tag sets and their corresponding presentation semantics are fixed; where as in XML their meaning is what the developer intends with few restrictions other than to meet basic formatting rules for the data tags. The separation of content and presentation proves to be a valuable characteristic associated with markup languages, enabling multiple display views for one set of data. For example, Figure 2 depicts a simplistic HTML file with scenario data presented to the end-user in an unstructured manner. Figure 3 demonstrates the same data represented in structured XML tags that allow the advantages of the markup language to be leveraged in presenting different views to the user.

```
<html>
        <head>
                <title>Scenario</title>
        </head>
        <body>
        <p> Scenario Properties: </p>
        <p> Location: Port Hueneme </p>
        <p> High Value Unit: DDG – 51 Arleigh Burke Class Destroyer </p>
                <p>Location: Inport Port Hueneme, California </p>
        </body>
</html>
```

Figure 2. Simple HTML Snippet with no structure to the data contained.

```
<scenario>
        <scenarioLocation> Port Hueneme, California
        </scenarioLocation>
        <highValueUnit type="DDG-51" description="Arleigh Burke Class
Destroyer" location="inport, Port Hueneme, California" />
</scenario>
```

Figure 3. Sample XML snippet with same information as in the unstructured HTML example but defined in a way that semantically significant data can be formally validated, accessed and manipulated.

E. EXTENSIBLE STYLESHEET LANGUAGE FOR TRANSFORMATIONS (XSLT)

The Extensible Stylesheet Language for Transformations (XSLT) is an XML based language that is used to apply standard formatting that can result in multiple views of an XML document. [Hunter 2001]. As Nicklaus showed with Marine Corps Operations Orders represented in XML, XSLT is the engine that can allow multiple representations or views to be created from the abstract data model represented by an instance of an XML document. [Nicklaus 2001] By programmatically applying sets of XSLT rules that are defined in XML against such an instance document, we can represent a tactical scenario in one instance while having many options to choose what visual representation of that document is applicable or appropriate to display to an end-user. For example, the hypothetical XML document shown in Figure 3 of this chapter can also be represented in 3D in addition to HTML through the application of XSLT stylesheets. (Figure 4)

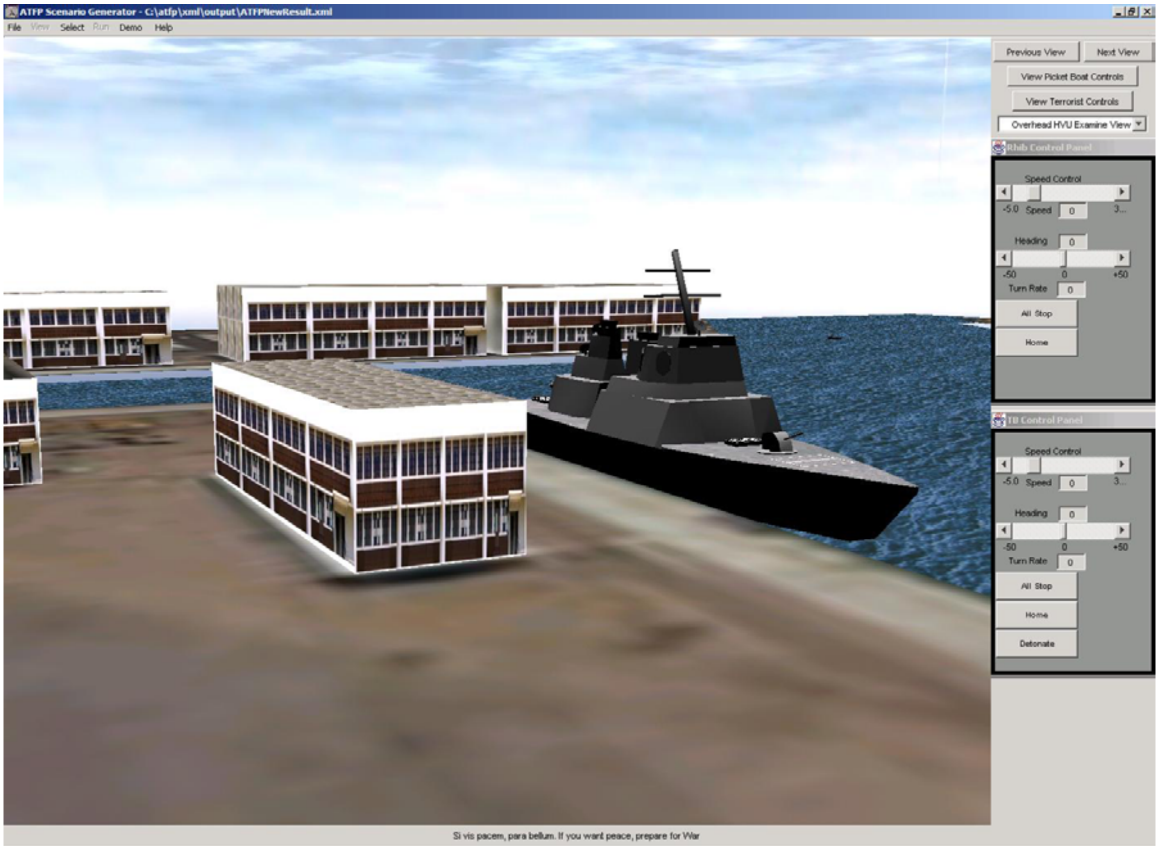


Figure 4. X3D Graphics Depiction of the Hypothetical XML Document from Figure 3.

F. DESIGN PATTERNS

As with many other Computer Science concepts, the original ideal of developing reusable solutions to recurring problems came from the field of architecture when Christopher Alexander wrote two books in the late 1970's that covered various patterns in building architecture and planning: *A Pattern Language: Towns, Buildings, Construction* (Oxford University Press, 1977) and *The Timeless Way of Building* (Oxford University Press, 1979).[Alexander 1977] [Alexander 1979] These ideals have been furthered within the software industry. For example, the *Patterns in Java* series of texts was studied both during the design and throughout the implementation of this thesis to identify applicable areas and reduce the amount of time in the invention of software design solutions that have been solved by others in the past. Although not an all-inclusive explanation of the design patterns utilized in the thesis, the following is a short

representation with explanation of the theory behind some of the patterns used in the execution of this thesis.

1. Mediator

Originally developed by the group commonly known as the 'Gang of Four' (GoF), Eric Gamma, Richard Helm, John Vlissides, and Ralph Johnson, the Mediator pattern uses one object to coordinate state changes between other objects. "Putting the logic in one object to manage state changes of other objects, results in a more cohesive implementation of the logic and decreased coupling between the other objects." [Grand 1998] This pattern allows a programmer to centrally manage complex object interactions that commonly occur in a User-Interface centric application, such as tactical scenario creation, configuration, and viewing, in a centralized location. With less dependency built into the many different classes and objects that are created in a large application development effort, more advanced levels can be reached quickly while maintaining an effective level of maintainability.

2. Observer

Also developed by GoF, the Observer pattern "allows objects to dynamically register dependencies between objects, so that an object will notify those objects that are dependent on it when state changes."[Grand 1998] This pattern is used heavily in the 'Listener' ideal that is central to reacting to user-driven actions in a Graphical User Interface environment. It lays the foundation for notification of events to objects that have registered an 'interest' in various state changes.

3. Singleton

Another original GoF pattern, the Singleton, "Ensures that only one instance of a class is created. All objects that use an instance of that class use the same instance." [Grand 1998] This concept is very useful when dealing with the central management of resources where there should be exactly one instance of the management class. For example, when using the Prototype pattern to generate copies of well-known objects to use in an application, the Prototype Builder or Manager could be instantiated utilizing the Singleton pattern. Also, in the context of this thesis, when creating the management

classes that coordinate the internal state of autonomous agents, the Singleton is again useful.

4. Delegation

"Delegation is a way to extend and reuse the functionality of a class by writing an additional class with added functionality that uses instances of the original class to provide the original functionality." [Grand 1998] The application of this pattern allows a programmer to more appropriately use the functionality of classes that might play one of many roles in a newly created class vice merely extending the use of one existing class. Autonomous agent programming implementations are often a good example of this pattern in practical use. Each agent or actor in a simulation can be comprised of numerous internal agents that all work together in different ways. Since the Java programming language allows for only a single inheritance hierarchy, we are faced with the difficult-to-manage case of cascading inheritance between classes, using multiple interfaces, or using the Delegation pattern to create a class with the functionality desired in a composite fashion.

5. Interface

The Interface pattern's intent is to "keep a class that uses data and services provided by instances of other classes independent of those classes by having it access those instances through an interface." [Grand 1998] The application of this pattern allows for runtime extensibility and binding of classes within an application, without having to know the inner details and workings of classes involved at compile time beyond the fact that they implement all public methods declared in the interface. Refer to [Grand 1998] for further details on the Interface Pattern design, or [Salles 2002] for further information on the implementation details and application for runtime-extensibility of this pattern within the NPSNET-V framework.

G. X3D GRAPHICS AND THE VIRTUAL REALITY MODELING LANGUAGE

X3D Graphics is the next-generation of the Virtual Reality Modeling Language 1997 (VRML97) 3D graphics format for the Web. In July, 2002, the final working draft of the recommendation for specification was released by the Web3D consortium

available online at www.web3d.org/specs (accessed March 2003) [X3D 2002]. Of note, X3D has been developed with an open-source sample implementation for specification implementation and evaluation along with support from major industry players interested in 3D content development for the Web. Since the format is XML based, it can also take advantage of the benefits of XML by using XSLT stylesheets to view the same content rendered in VRML97, HTML, or with direct rendering of the XML-based tree structure in the open-source browser implementation, Xj3D. Theses [Hunsberger 2001] and [Nicklaus 2001], along with examination of the X3D specification reference above, provide an in-depth overview of most of the over 130 defined nodes within the X3D scene-graph structure. More in-depth explanation of syntax will be provided in context as required in the remainder of this thesis.

H. SCALABLE VECTOR GRAPHICS (SVG)

In the late 1990's, the same need was identified for an open-standard interchange format for 2-Dimensional (2D) graphics as was identified for 3D graphics. Since many graphic design programs rely upon proprietary formats, there was no common way to easily exchange information. As a result, similar to the Web3D Consortium's efforts with VRML and now X3D, the World Wide Web Consortium created a working group for the development of vector graphics as an XML application, now referred to as Scalable Vector Graphics (SVG). [Eisenberg 2002]

Similar to X3D for 3D, SVG keeps the 2D graphical views of data separate from their abstract data model. One difference, though, as depicted in Figures 5-7, is the common 2D images that have been supported by Web browsers since the early 1990's have only supported raster formatted 2D graphics which lose image quality when they are scaled. For example, when one zooms in on a Joint Photographic Experts Group (jpeg), Portable Network Graphics (.png), or Graphics Interchange Format (gif) image, the pixilation artifacts can be severe. The application currently being used to view the image must calculate a uniform manner in which to expand the image properties associated with each pixel of the image which results in a reduction of image quality. [Eisenberg 2002]

As depicted in Figures 5 and 6, with a standard, open-based vector format such as SVG one can scale without loss of quality of the original image.

Figure 5. Original View of Batik3d.svg from http://xml.apache.org/batik/

(accessed January 2003)

Figure 6. Example of maintaining image quality while zooming in on Batik3D.svg in the Adobe SVG Viewer plug-in loaded in Internet Explorer 6.0.

Figure 7. Example of Loss of Quality While zooming-in on a rasterized version of the batik3d image (png depicted).

Additionally, one benefits from the same advantages found with a common interchange format with 2D graphics as with 3D graphics through use of XSLT for conversions. The same markup language document can be utilized in both 2D with SVG as well as 3D with X3D. This technique is demonstrated later in this thesis with statistical data from AT/FP scenarios.

I. THE JAVA PROGRAMMING LANGUAGE

The primary programming language during the execution of this thesis was the Java Programming Language by Sun Microsystems. As stated recently by [Salles 2002] and explained in greater detail on the Sun Microsystems Java website at http://java.sun.com (accessed March 2003). Java is a Web-friendly language that is cross-platform capable at run-time without having to recompile. Java is the primary language of choice for openly repeatable network-capable interoperable programs.

1. JAVA 3D

Java 3D is a scene-graph based 3D Application Programming Interface (API) provided by Sun. It is utilized as one of the rendering engines behind both the NPSNET-V and open-source Xj3D engines also used for this thesis. Java 3D programming is not required for X3D rendering. It can be used directly to augment Xj3D for special effects if desired.

2. JDOM

JDOM is a Sun approved extension to the Java 2 API for using XML to process data. More information for the details of JDOM use can be found at http://www.jdom.org (accessed February 2003).

J. X3D, ECMASCRIPT, AND THE JAVA PROGRAMMING LANGUAGE

The X3D Script node allows models, controls, or dynamic graphics creation to be incorporated into a Web-based 3D scene with programming languages such as Ecmascript, Visual Basic, or the Java Programming language. [X3DSPEC 2002] The Script node can also be used in conjunction with other nodes defined within a scene to create more complex behaviors, and to provide functionality such as network access with

the IEEE Distributed Interactive Simulation (DIS) X3D Profile, physics models, or web-based graphical user-interfaces. [Hunsberger 2001] This is also a means by which software developers can further extend the X3D specification such as Yumetech Inc.'s demonstration of a Multi Texture node that wasn't part of the initial X3D specification recommendation.

K. DIS-JAVA-VRML

DIS-Java-VRML is another open-source API and collection of exemplar implementations that demonstrate the use of the X3D/VRML97 scripting node to allow Large Scale Virtual Environments (LSVE's) to be run within a standard web-browser such as Netscape Navigator with freely available commercial plug-ins. Since DIS is an International Standards Organization (ISO) standard and a well-understood protocol, it provides a common framework for geographically separated collaborators or researchers to communicate over the World-Wide-Web. [Hunsberger 2001] More information and examples on DIS-Java-VRML can be obtained at: http://www.web3d.org/WorkingGroups/vrtp/dis-java-vrml. (accessed March 2003)

L. XJ3D OPEN SOURCE PROJECT

Xj3D is the open-source rendering implementation for the X3D graphics standard. It is "a Java-based toolkit developed by Yumetech that allows companies to rapidly support X3D."[X3D 2002] The Web3D Consortium has also formed the Java Rendering Working Group consisting of members from Anaviza Inc., Sun Microsystems, and Yumetech that are concurrently working on the definition and implementation of bindings for various common graphical API's such as OpenGL® and Direct3D™. Upon completion, this implementation will make the specific graphics rendering context of X3D graphics agnostic to the commercial up's and down's of the market place or consumer popularity.

M. NPSNET-V

NPSNET-V is an on going Naval Postgraduate School research project that is a Run-Time Extensible Virtual Environment framework. [Salles 2002] NPSNET-V is utilized as one of the available 3D viewing frameworks for scenarios generated using X3D graphics definitions and rendered with the Xj3D code-base within a Java3D environment. [Salles 2002] or the project website at can be referred. to for more information (http://www.sourceforge.net/projects/npsnetv/). (accessed March 2003)

N. AGENT-BASED SIMULATION

To complement the warfighter's subject matter knowledge in any warfare area, in the case of this thesis Anti-Terrorism/Force Protection, agent-based simulation provides a valuable tool for playing "what-if" scenarios to examine the tactical implementation of military doctrine. Agent-based simulation refers to modeling entities that have internal intent and the ability for effecting autonomous action upon the simulation environment. [DICKIE 2002] With simulation of the asymmetrical threat being difficult to do, agent-based simulation affords a promising avenue in which to gain insights to potential shortcomings in the tactical implementation of Anti-Terrorist defensive plans.

1. Multi-Agent Systems

The primary definition for a Multi-Agent System utilized in the scope of this thesis is from [FERBER 1999]. He states that a multi-agent system is "an electronic or computing model made up of artificial entities which communicate with each other and act in an environment." [Ferber 1999] He goes on to express this ideal in a more formal manner shown in Figure 8.

MAS = {E, O, A, R, Op, Laws}

E – Environment
O – Objects situated in the environment
A – Agents, $(A \subseteq O)$
R – Relations linking objects O
Op – Operations
Laws – Constraints governing the environment

Figure 8. Multi-Agent System originally from [Ferber 1999] and [Osborne 2002]

A more in-depth overview of the various existing multi-agent system definitions can be found in [Osborne 2002]

2. NPS-Developed Composite-Agent Architecture

In order to leverage strengths of both cognitive and reactive agents within an environment while simplifying the design through an indirect utilization of the Delegation Pattern mentioned above, the Composite Agent (CA) architecture (Figure 9) has been developed and implemented in several different contexts at the Naval Postgraduate School MOVES Institute. [Osborne 2002] Modeled upon a common robotics paradigm, “Sense-Decide-Act”, and more formally applied to autonomous agents by [WOOL 2002] as “observe – update state – act”, a composite agent is defined to be comprised of Symbolic Constructor Agents (SCAs) that sense their outer environment to aid in the creation of an inner environment which can be acted upon by a set of Reactive Agents (RAs) who are each “responsible for promoting a specific behavior of the Composite Agent” [Osborne 2002]. This thesis builds upon the architecture developed by [Osborne 2002] in the CA concept in order to explore the possibilities of modeling cognitive templates within the given architecture.

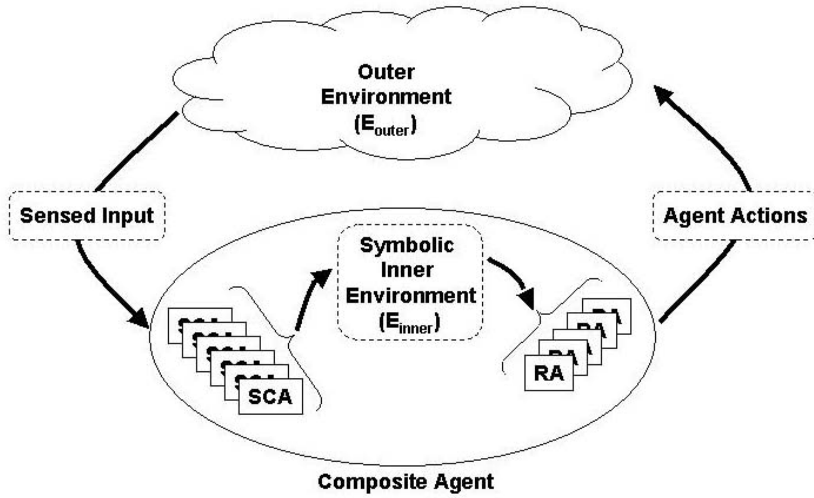


Figure 9. Composite agent architecture from [Osborne 2002]

O. EXTENSIBLE MODELING AND SIMULATION FRAMEWORK (XMSF)

The Extensible Modeling and Simulation Framework (XMSF) is defined as a set of Web-based technologies, applied in an extensible framework, that enable a new generation of modeling and simulation (M & S) applications to develop, emerge, and interoperate. [XMSF 2002] The primary subject areas for XMSF consist of: 1) Web/XML, 2) Internet/networking, and 3) Modeling and Simulation (M & S). The principal institutions which are leading the research and development of XMSF are: 1) Naval Postgraduate School (NPS) Moves Institute, 2) George Mason University (GMU) NetLab, Science Applications International Corporation (SAIC), and Old Dominion University (ODU) Virginia Modeling, Analysis & Simulation Center (VMASC) Battle Lab. [XMSF 2002a]

P. SUMMARY

This chapter describes the high level concepts of the many technologies explored and exploited to provide a contextual understanding for the remainder of the thesis. If more information and a greater in-depth understanding, the reader is referred to the List of References as well as those mentioned above.

III. OVERVIEW OF PROBLEM

A. INTRODUCTION

Currently, there is little to no M&S technology deployed for the tactical warfighter to play 'what-if's' for the planning and deployment of tactical forces while following established doctrine. Typical roles of M&S technology at the tactical level of war are to aid the operator in improving proficiency in the use of equipment or simulating war-time scenarios with the equipment, but not geared to the planning stage with statistical analysis and visualization provided in order to improve outcomes in the field. Taking advantage of this opportunity, this thesis demonstrates how to take a subset of the many AT/FP situations faced by today's navy against the surface-borne threat. It also demonstrates how web-based technologies can be utilized to aid the warfighter in the implementation of tactical-level plans by dynamically defining scenarios representing potential defensive plans against terrorists. Then, it provides various options for running 'what-if' scenarios on those plans. The idea of being able to run these types of scenarios was originally posed for web-based modeling and simulation utilizing X3D graphics by [Murray 2000] with the auto-generation of visualization for Air Tasking Orders.

B. PROBLEM STATEMENT

Our problem is to first identify what is required to leverage M&S technology to demonstrate a repeatable process that can be applied as providing tactical level tools to the warfighter to better employ published doctrine. Specifically, in the war on terrorism for planning against the defense of high value naval units against the surface-borne terrorist threat. Then, to provide a prototypical implementation for a limited number of locales and to show how it may be extended to provide insight against other threats.

C. PROPOSED SOLUTION AND RESEARCH FOCUS

The proposed solution to the problem stated above incorporates many of the various ideals put forth in the last few years in web-based M&S technologies, while developing new methodologies and content where applicable. Central to the solution is

the representation of models and views within to allow content to be manipulated and viewed in different formats separate from the model definition and representation.

With XML work it is common to create exemplar components and views before embarking on a specified local definition for representing what one cares about, in our case tactical scenarios for AT/FP doctrine. So, the initial focus of the research and problem solution was on development of the components necessary to represent a tactical scenario. This began with the authoring of a surface ship not currently in the Scenario Authoring and Visualization for Advanced Graphics Environments (SAVAGE) content library hosted at NPS at http://web.nps.navy.mil/~brutzman/Savage/contents.html. (accessed March 2003) Next, further components, and basic physical interactions between entities within a virtual world were explored in the development of an unclassified, web-based reconstruction of the terrorist attack on the USS COLE http://web.nps.navy.mil/~brutzman/Savage/Scenarios/UssColeTerroristAttack/UssColeTerroristAttack.wrl . (accessed March 2003)

Then, we shifted gears a bit and investigated a 'user-centric' design process for implementation of a suitable user-interface for the end-user that does not have intimate development-level knowledge of the various web-based technologies utilized, and showed how that process was applied in the context of this thesis independent of internal development efforts. Further, we defined our tactical scenario high-level model representation, and incorporated it into an XML schema definition. Following the schema definition, the explanation of how XML stylesheets were used to transform a locally defined scenario definition into several different views that can be utilized by the end-user in unique and interesting ways. We then defined the user controls and physical models incorporated and investigate what information from the real world needs to be represented in the model and what is abstracted away for the context of our problem. Following this, the autonomous agent design considerations and implementation were defined and implemented to aid in gaining insight for our defensive plan both visually as well as over multiple-scenario replications in order to see if any 'outlier' scenario possibilities and behaviors can be identified. Finally, a use-case of the work developed was demonstrated as well as further application of the ideals presented in this thesis and

how they can be applied to further military education, training, and experimentation in a rapid, dynamic, web-based manner.

D. DESIGN CONSIDERATIONS

With the wide-spread deployment of IT-21 to the U.S. surface navy enabling internet connectivity whilst deployed to the far-reaches of the planet for both unclassified and classified internets, our model designs will need to incorporate the least amount of data required to help answer our research question in order to provide a context in which this or other future work can be deployed for training of operators at sea today. Just during the work on this thesis, Central Processing Unit (CPU) speed for personal computers has jumped from 867 MHz to 2.6 GHz in the span of a little over one year as well as the further deployment of high-speed Internet connectivity. It is expected that constraints that apply in the design and implementation of this context should be examined on a regular basis for all web-based research and deployment of services to continue to provide the quality of service expected by end-users.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. DEVELOPING SCENE COMPONENTS

A. INTRODUCTION

This chapter will demonstrate how scenario components are developed for representing the 3D view of our models. It then demonstrates how a reconstructive scenario can be created in order to view and gain insight from past events, providing a recreation of the attack on the USS COLE that occurred in October 2000. Further, it reviews the real-world computational physics utilized for dynamic scenario interaction and control of entities. Then, it briefly examines the advantages and disadvantages for various networking options available for these purposes, demonstrating an effective problem solution. The chapter concludes by examining other physics-based subjects worthy of more in-depth explanation.

B. DESIGN AND IMPLEMENTATION OF THE DDG-51 X3D MODEL

The DDG-51, Arleigh Burke Class Destroyer X3D model was the first graphics project undertaken for this thesis. The process of model design and creation without the benefit of design plans or CAD data follows the following steps:

1) Acquire suitable unclassified and open resources on which to base one's implementation. For military models, an excellent resource is the Federation of American Scientists' (FAS) website at http://www.fas.org (accessed March 2003) Specifically, the primary source of information for the Arleigh Burke model is collected from: http://www.fas.org/man/dod-101/sys/ship/ddg-51.htm. (Figure 10) (accessed March 2003) This website provides unclassified pictorial representations of various military vessels, as well as detailed ship specifications which can be leveraged in model implementation.

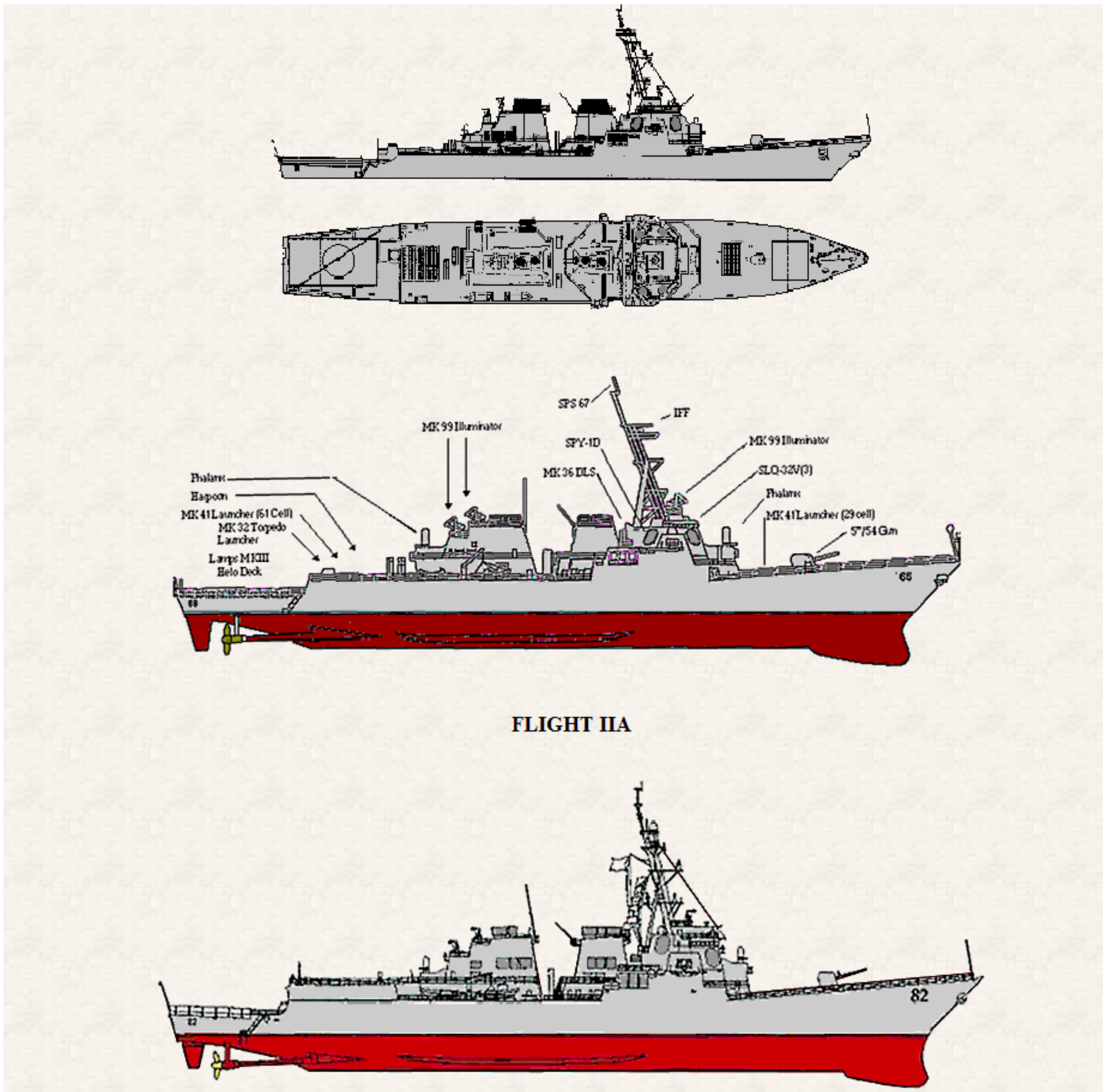


Figure 10. Depiction of DDG-51, Arleigh Burke class destroyer from the FAS web site http://www.fas.org/man/dod-101/sys/ship/ddg-51.htm. (accessed March 2003)

2) Once a suitable data resource is found, the next step is **to break the model into as many logical components as possible for 3D creation.** In the case of the DDG, without the advantage (at the time) of authoring of a previous DDG model or availability of CAD data to work from, it was found easiest to progress in a bottom-up fashion starting with the ship's screws and shafting. Unlike other basic components such as guns

and basic hull forms, not many exemplars of either civilian water-craft propellers or military ship screws were found. As a result, alternate resources had to be found in order to have a basis on which to create these components for use with the Arleigh Burke model. Two sets of data were found that proved to be suitable, the U.S. Naval Institute Proceedings (Figure 11) and the USS ROSS (DDG 71) Commissioning Book from June 1997.

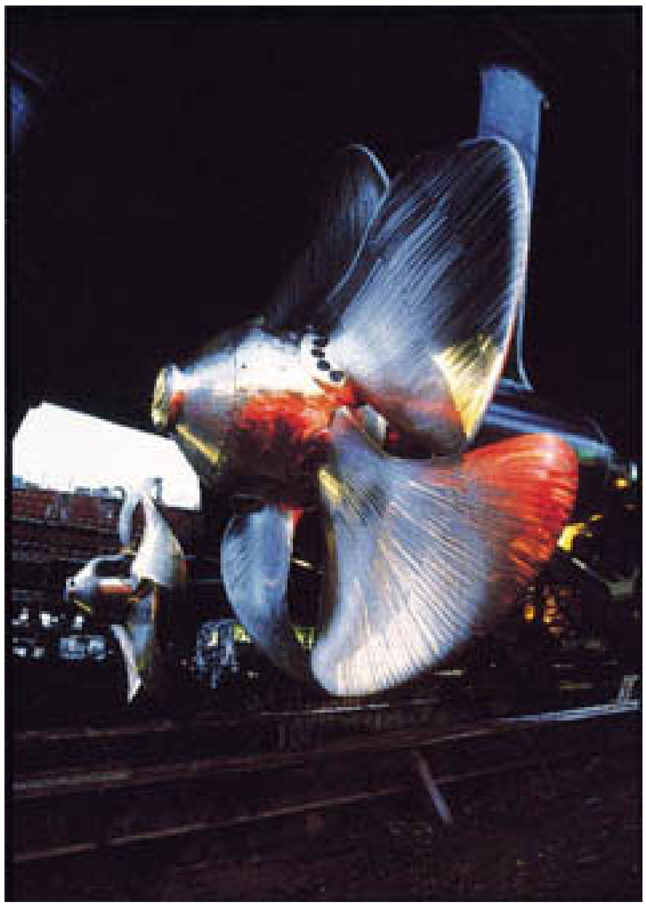


Figure 11. Depicts one of two images utilized for 3D creation of Propellers and Shafting for the DDG-51 X3D model from http://www.usni.org [USNI 2001] (accessed March 2003)

In this case, the procedure is to first reproduce the propeller hubs and shafts, followed by one blade which is then repeated using “copy-by-reference” (i.e. DEF/USE constructs) in the geometry definition. (Figure 12)

```
<Transform rotation='1 0 0 -1.64' containerField='children'>

    <Group DEF='BLADE' containerField='children'>

        <Transform rotation='0 0 1 -.349' scale='1 .1 1' translation='-2 2.25 0' containerField='children'>

            <Shape containerField='children'>

                <Appearance containerField='appearance'>

                    <Material USE='BRASS' containerField='material'/>

                </Appearance>
                <Extrusion creaseAngle='1.1' crossSection='2, .2, 1.95 .5, 1.9 1, 1.95 1.5, 2 2, 2.1
                1.5, 2.1 3, 2 3.5, 1.75 4, 1.65 4.5, .5 4.85, .4 5, .5 5.4, 1 5.6, 1.5 5.9, 2 6, 2.5 6.05, 3
                6.05, 3.5 6.05, 4 6.05, 4.5 6, 5 5.95, 5.5 5.6, 5.6 5.5, 6 5.25, 6.5 5, 6.8 4.5, 7.1 4, 7.4
                3.5, 7.45 3, 7.4 2.5, 7.15 2, 7 1.5, 6.5 1.25, 6 1, 5.5 .95, 5 .925, 4.5 .85, 4 .55, 4 .45,
                3.5 .46, 3 .5, 2.5 .46, 2 .2' containerField='geometry' spine='0 0 0 0 1 0'/>

            </Shape>

        </Transform>

    </Group>

</Transform>
<Transform rotation='1 0 0 -1.64' containerField='children'>

    <!-- Approx 72 degrees of sep between the center of each blade as it touches the hub -->
    <Transform rotation='0 1 0 1.36' containerField='children'>

        <Group USE='BLADE' containerField='children'/>

    </Transform>
```

Figure 12. Depicts the X3D definition of 1 propeller blade and an example of the re-use of the single definition.

The final result of the visual model of the Arleigh Burke Class Destroyer propellers was completed in approximately one week of non-full time work and is depicted in Figure 13.

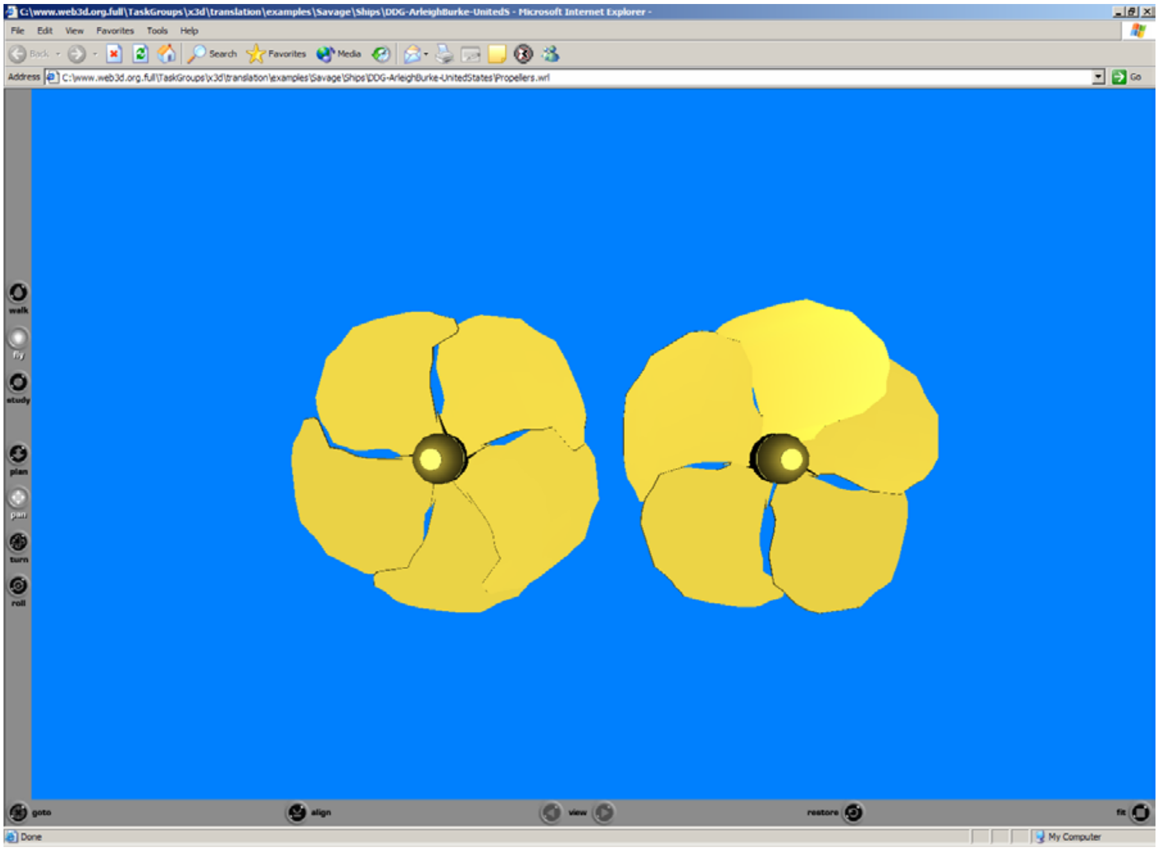


Figure 13. Screen snapshot of Propellers.wrl rendered in the ParallelGrahics Cortona VRML97 client plugin in Internet Explorer v6.0. File available online at http://web.nps.navy.mil/~brutzman/Savage/Ships/DDG-ArlieghBurke-UnitedStates/Propellers.wrl (accessed February 2003)

The hull was modeled next, in the same fashion as the propellers model, followed by the flight deck as a separate component and continuing through the remainder of the ship

3) The next step while developing all scenario components is to decide **what level of detail is to be modeled.** Pertinent issues contained within level of detail are: a) amount and level of detail of applying texturing to models, b) amount of lighting and effects to apply, and c) the level of fidelity of sub-components to model. In the construction of the DDG-51 model, basic texturing was completed for flight deck and fore-castle markings, and the flight deck netting. (Figure 14)

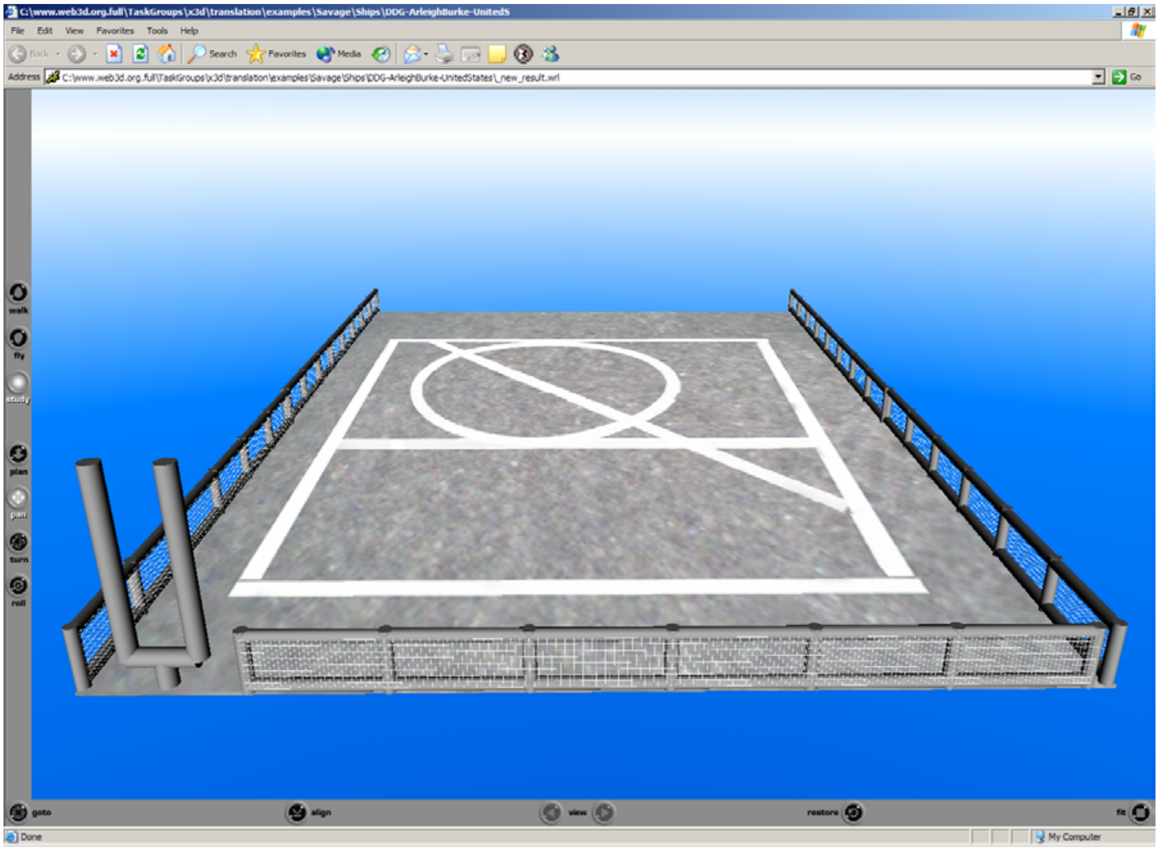


Figure 14. FlightDeck.wrl component from the Arleigh Burke Class X3D model.

The default lighting model was chosen for use, and the current level of fidelity for this implementation did not require specific ship decking components to be modeled, only sufficient detail to permit possible helicopter operations. Further detail on decking and access ports will be required for possible future work with integration of multiple individual humanoid models.

.

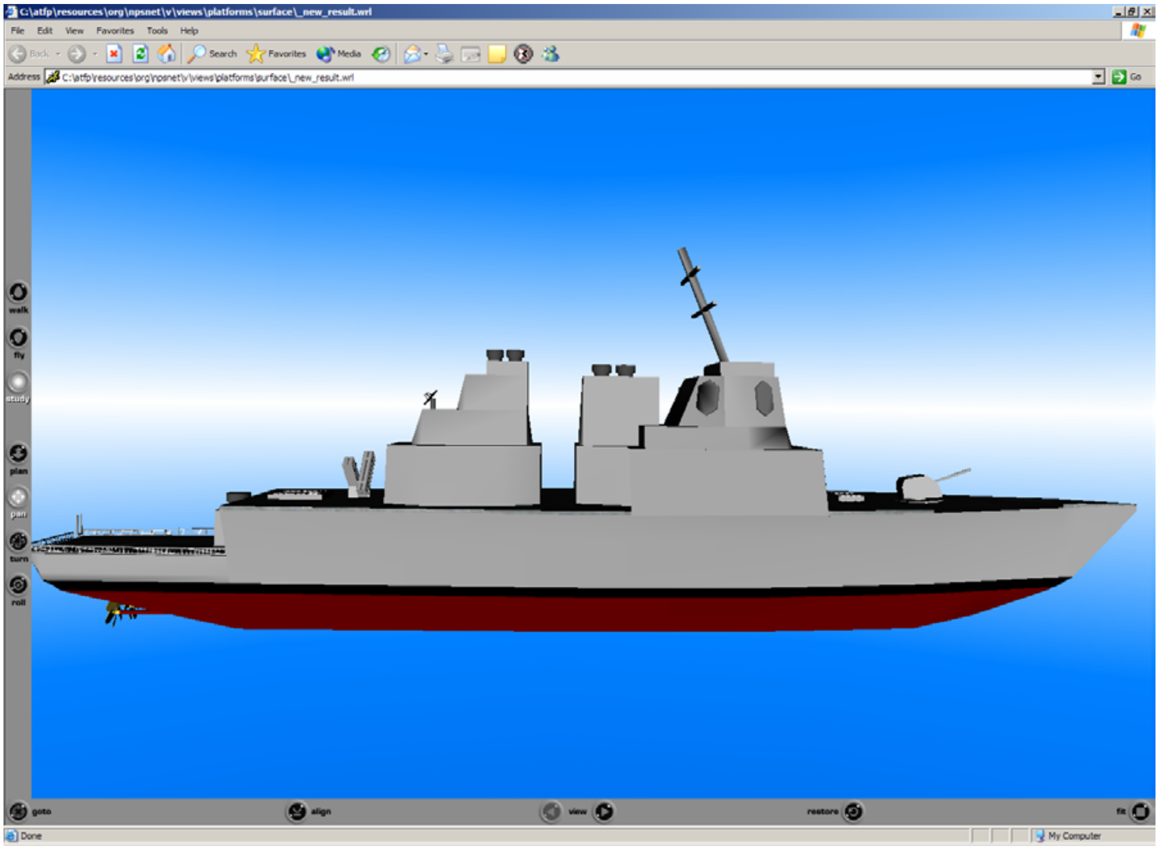


Figure 15. Screen Snapshot of the complete Arleigh Burke Class X3D Model.

Once the necessary X3D techniques were mastered so that the initial modeling of the DDG-51 was complete (Figure 15), further ship models were rapidly prototyped and produced for planned use in larger contexts. For example, the standard navy rigid-hull-inflatable-boat (RHIB), Figure 16, was developed in the span of one evening with the experience of the Arleigh Burke model creation completed.

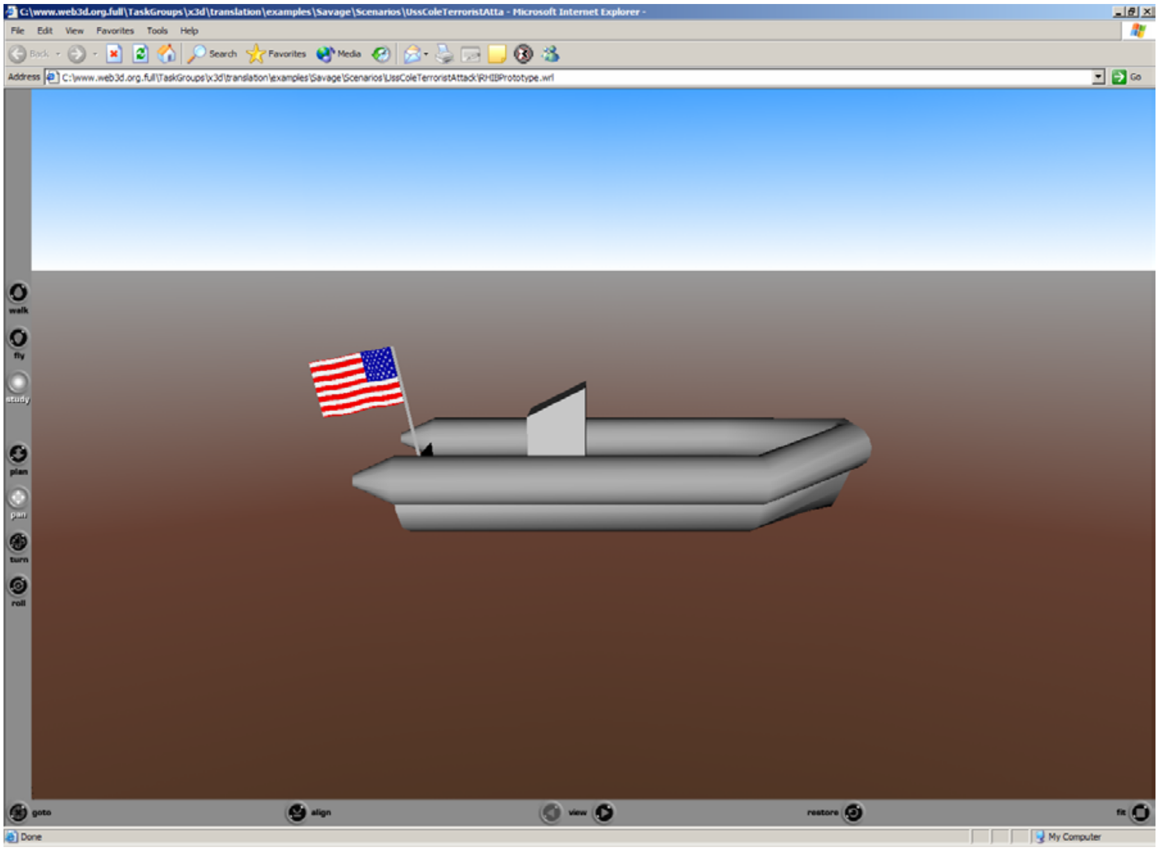


Figure 16. Screen Capture of the RHIBPrototype.wrl developed for defensive scenario. Length 6.8 meters.

C. DESIGN AND IMPLEMENTATION OF THE USS COLE TERRORISTATTACK X3D MODEL

After implementation of X3D components, the next step was development of a reconstruction scene to allow the development of a graphical framework in which to develop further work. The terrorist attack on the USS COLE (DDG 67) that occurred October 2000 in Aden Harbor, Yemen was chosen for this purpose. Modeling steps for this process were: 1) Landscape Creation, 2) Further Entity Development, 3) Entity-Entity Interactions, 4) Unclassified Weapons Modeling, 5) Real-Time Track Reconstruction, and 6) Dynamic Play back of the Scenario. These steps are next explained in greater detail in the remainder of this chapter.

1. Constructing the Geography and Pier

As first mentioned in [Blais 2002b], it was determined that the level of detail provided by Digital Terrain Elevation Data (DTED) Level 1, which contains terrain

elevation postings at approximately 100 meter intervals, suffices for the representation of Aden Harbor and the surrounding vicinity for the purposes of reconstructing the attack on the USS COLE since there are large mountains in the vicinity of the harbor. Furthermore, the refueling dolphin being modeled is not located alongside the shore, but rather in the center of the harbor. The methodology utilized in this creation of the Yemen locale was to modify a MatLab script adopted by [Blais 2001] in the creation of a Camp Pendleton (Figure 17), California scenario. The resulting Aden Harbor area is depicted in (Figure 18).

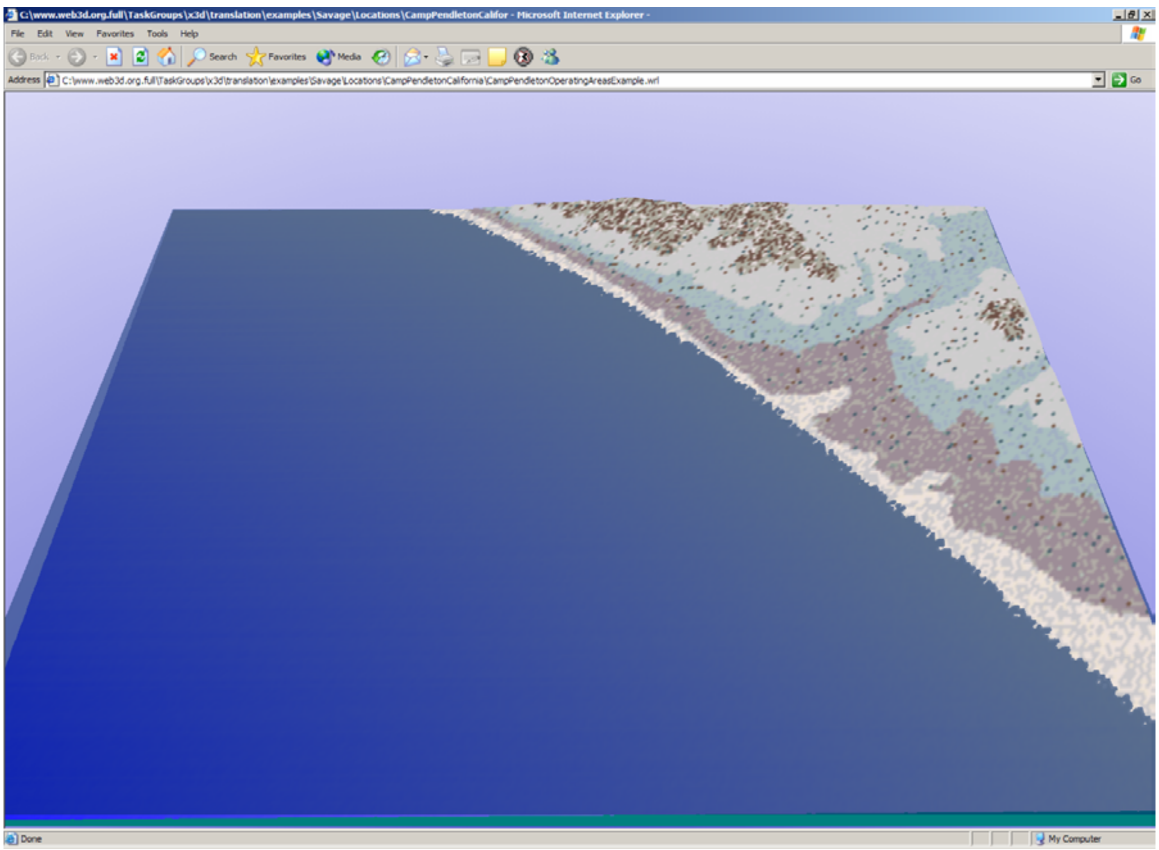


Figure 17. Screen Snapshot of the Camp Pendleton Geography created by the MatLab script and modified for creation of Aden Harbor, Yemen. Available online at: http://web.nps.navy.mil/~brutzman/Savage/Locations/CampPendletonCalifornia/CampPendletonOperatingAreasExample.wrl (accessed March 2003)

While investigating the requirements for re-use of the previously developed scripts for Camp Pendleton, it was discovered that besides Eastern Hemisphere Longitude markings needing to have a negative factor applied while using MatLab's command line features, it

is also necessary to do basic testing on each version of DTED CD's utilized to see what height level represented sea-level. In some versions this was indicated by 0 and other versions by -1 with the DTED specification silent on what the correct value is [DTED 1996].

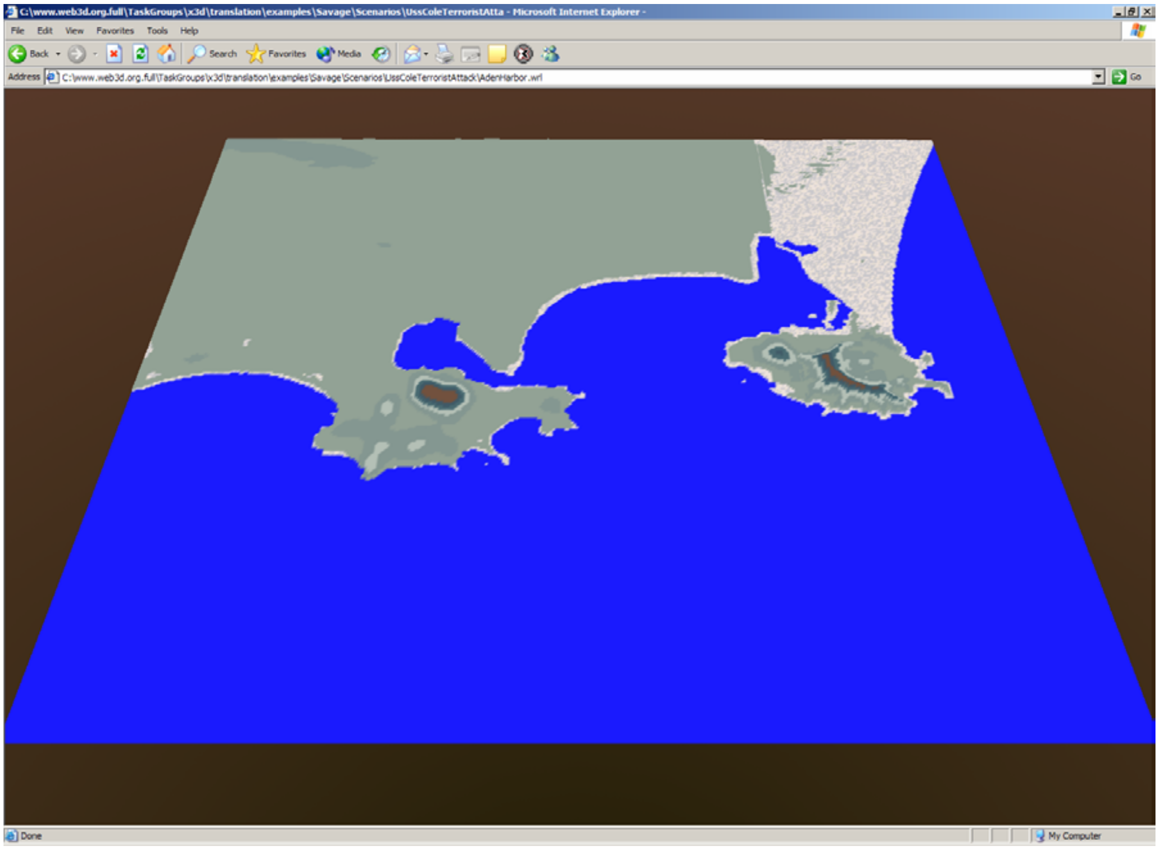


Figure 18. Screen capture of Aden Harbor, Yemen X3D scene utilized for reconstruction of the terrorist attack on the USS COLE (DDG 67). Available online at: http://web.nps.navy.mil/~brutzman/Savage/Scenarios/UssColeTerroristAttack/AdenHarbor.wrl (accessed February 2003)

Once the surrounding landscape was completed, it was then necessary to model the refueling pier where COLE was moored while attacked. Recreation of refueling dolphin 7 proved to be fairly straightforward with the use of the Web3D.org provided open source tool for scene graph and geometry creation, X3D-Edit. X3D-Edit is a graphics file editor for X3D that enables the XML tree of an X3D file to be edited in an environment that checks correct syntax and quickly generates the 3D view of authored

changes. With the use of reference photos and this editor, a fairly realistic depiction of the pier was quickly created (Figures 19, 20, and 21).

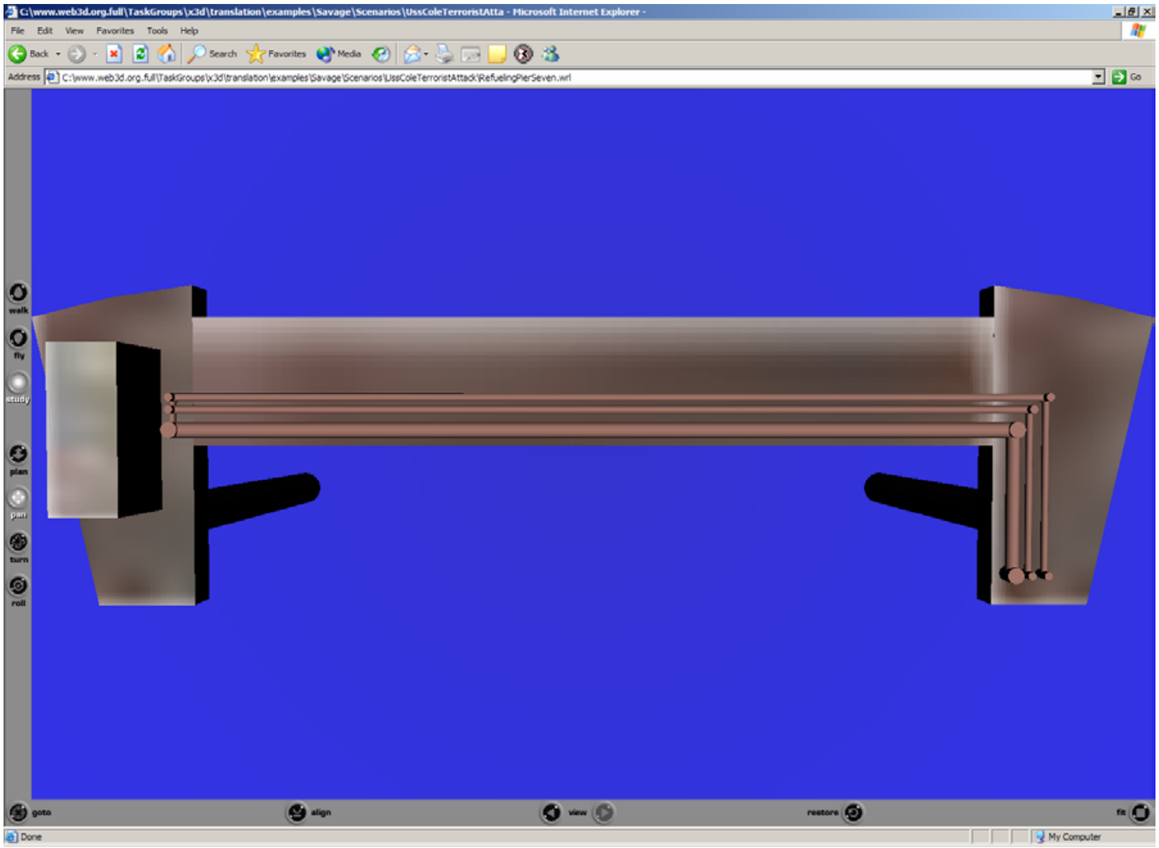


Figure 19. Overhead view of Refueling Dolphin 7 from the USS Cole Terrorist Attack scene. Available online at: http://web.nps.navy.mil/~brutzman/Scenarios/UssColeTerroristAttack/ RefuelingPierSeven.wrl (accessed February 2003)

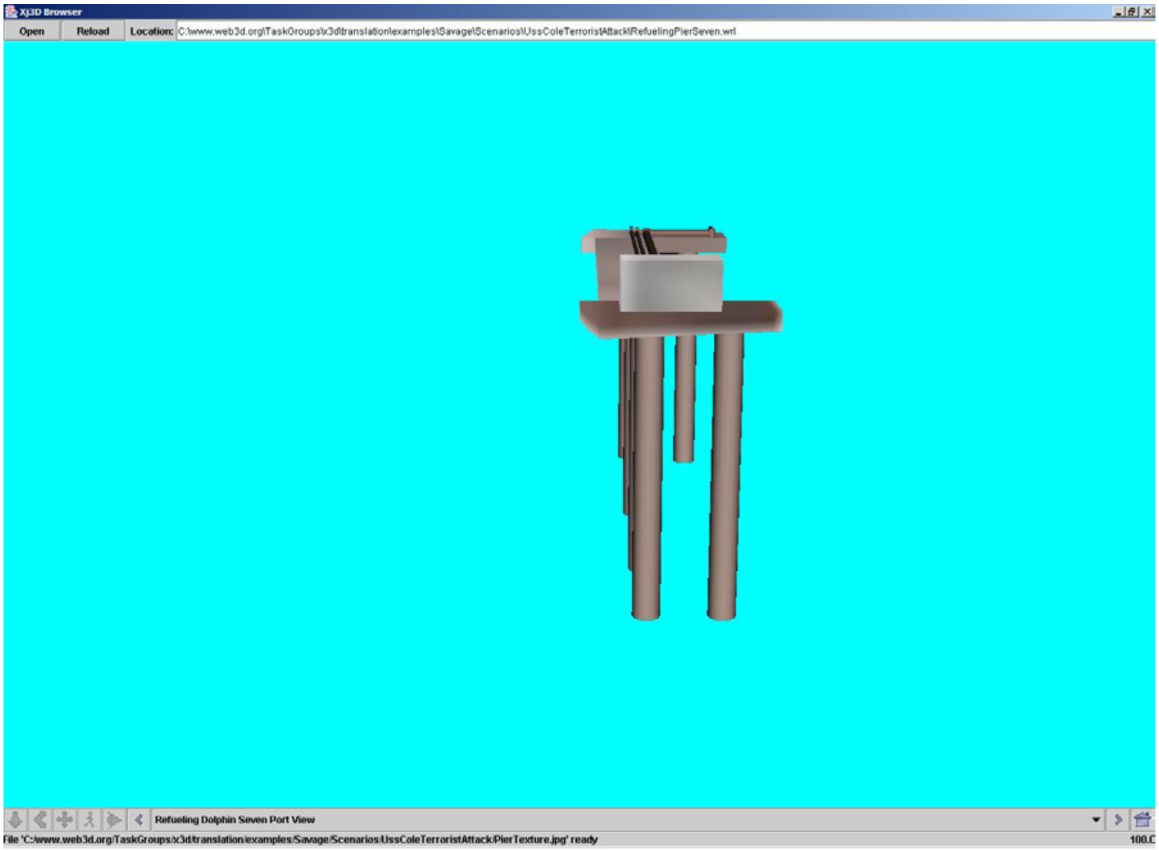


Figure20. Port View of the Refueling Dolphin at Aden Harbor.

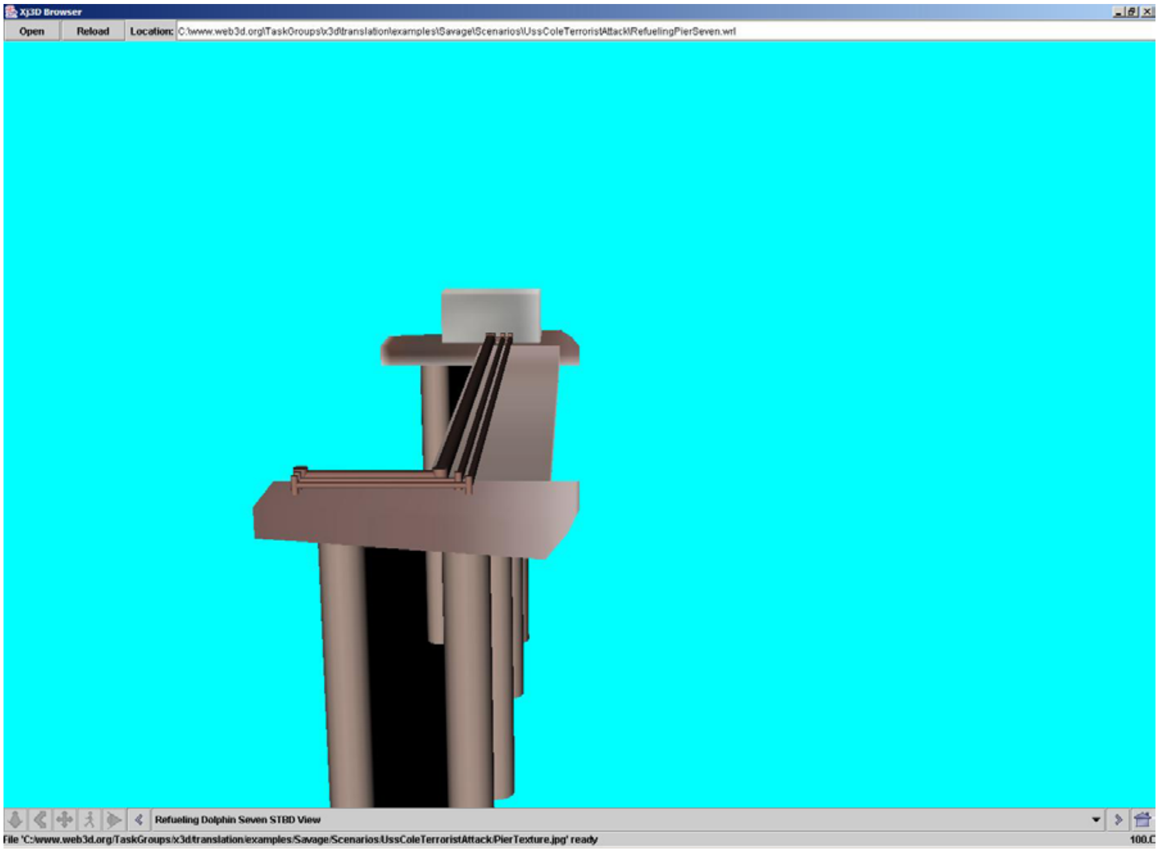


Figure21. Starboard oblique view of the refueling dolphin at Aden Harbor.

2. Entity Construction

After the landscape for Yemen was completed, it was necessary to further develop or identify other existing SAVAGE repository models to utilize within the recreation of the attack. As mentioned in [Blais 2002b], it was decided early in the modeling process that creating models that have the simple but effective "look and feel" was sufficient, investing more time towards accuracy of event timelines and entity behaviors than trying to obtain "photo-realism". To this end, low-resolution conceptual models (Figure 20) were created for the various small-craft referenced in the Court Of Inquiry into the attack on the Cole, without providing reference photography.

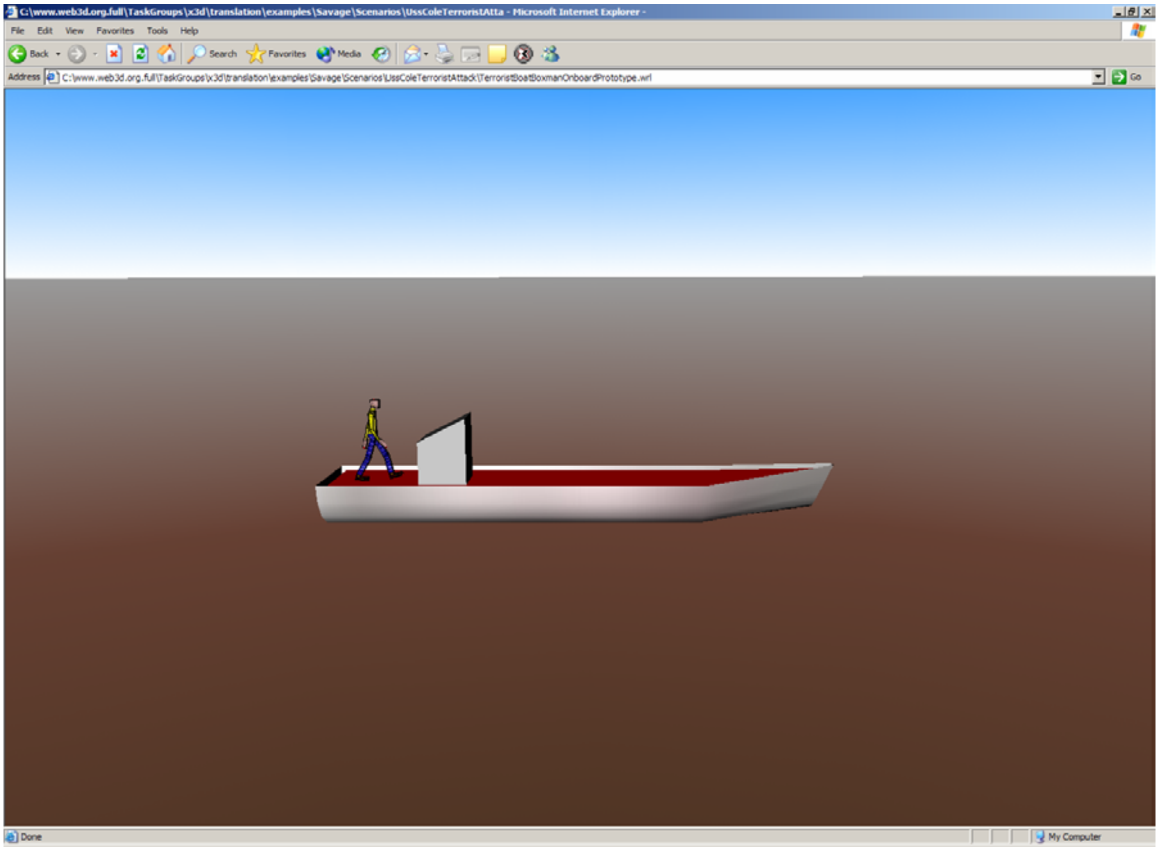


Figure 22. Conceptual View of the Terrorist Boat used to attack the COLE with the Boxman.wrl humanoid used as the ship driver. Drawn to scale (length 10.7 meters overall)

3. Explosion Modeling

For modeling the explosions in the context of this reconstruction, the U.S. Army's unclassified TNT equivalency model (Table 1) was utilized to represent the various ranges of damage that occur [USNA 1998]. An unclassified failure rate for steel was used to determine three levels of damage to represent in the 'scientific' view of the explosion being rendered: 1) structural Failure, 2) severe Damage, and 3) light damage. Each of these ranges was then represented through the use of different colored spheres that allow an analyst or developer to get a visual indication of what the real effects of an explosion might be, or in the case of the COLE see how closely they match the actual events.

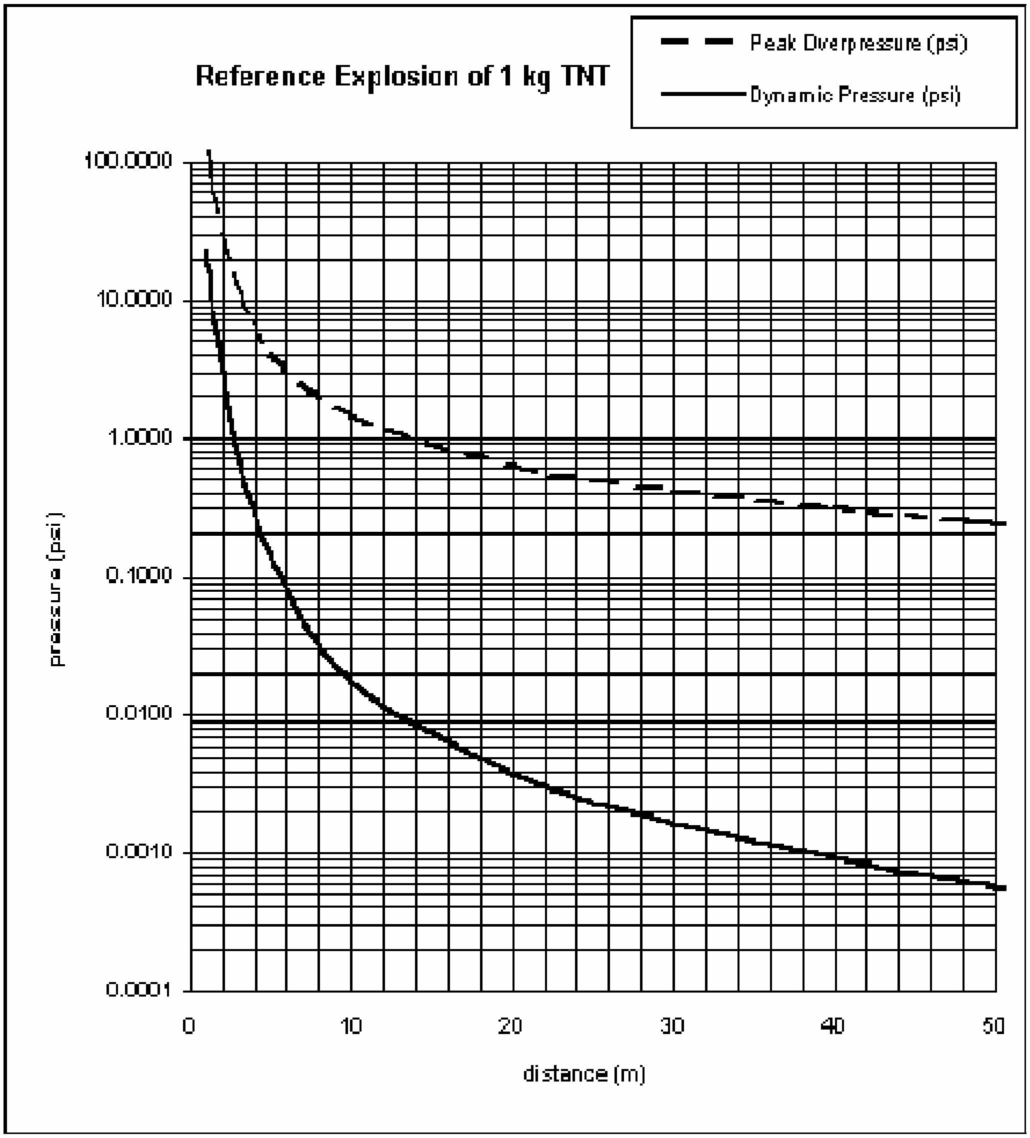


Table 1. U.S. Army TNT Equivalency Model from http://www.fas.org (accessed January 2003)

Unclassified reports put the amount of explosive material onboard the terrorist skiff that attacked the COLE at approximately 1/3 tons of TNT equivalent, based on the reported size of the boat. The TNT scaling law ($d_w = d_0 W^{1/3}$, where d_0 is the distance from 1KG of TNT, d_w is the distance from W KG of TNT equivalent, and W = equivalent amount of TNT) was then used in conjunction with Table 1 and theoretical steel failure rates to determine the conceptual distances at which to plot the different spheres in the X3D explosion model. When compared to a texture of the actual blast damage drawn to

scale on the Arleigh Burke Class X3D model (Figures 23 and 24), it was found that the approximate size of the blast spheres favorably corresponded with that of the actual damage. Thus, it was not necessary to divert resources into a higher-resolution physics API for calculating explosion results.

Figure 23. Picture depicting damage to the COLE after the terrorist attack [JMOCN 2002]

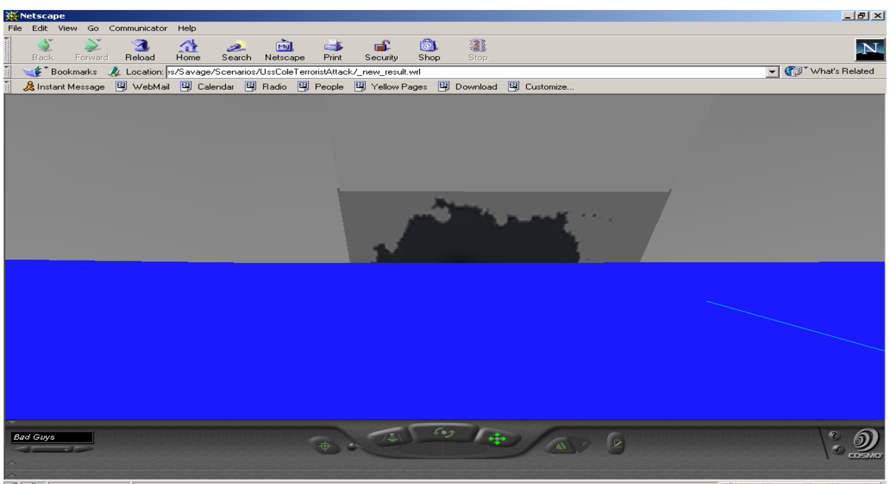


Figure 24. Screen capture of the damage depicting in the X3D Reconstruction of the attack on the 3D Model of the COLE. Shown at same scale as photograph in Figure 23.

4. Entity Track Animation

The realistic animation of entities of interest in any scenario reconstruction can prove to be difficult. It is mathematically tedious to smoothly represent 3D positions along with smooth, yet accurate animation through turns and reactions with the environment. Interaction among multiple entities is further challenging, especially as the number of entities in a scenario increases. A prototyping nook previously developed by the SAVAGE project team that allows for more rapid scenario development is the Waypoint Interpolator prototype. This prototype is utilized in a traditional object-oriented fashion in order to drive the tracks of each entity within the scene over a four-hour timeframe. Since the required kinematics mathematics required are encapsulated in a 'black-box' like manner within Ecmascript inside the prototype while remaining both editable and viewable to see what is occurring by scene authors, a scenario designer only has to provide speeds, times, positions, depth, and course for an entity to be precisely maneuvered within a scene. The time savings is great. Whereas during the construction of the Arleigh Burke class ship model it took over 12 hours to simply animate three missiles and one ship, greater numbers of entities were animated in more detail in the reconstruction of the COLE attack in less than one-third of the time. Key events in the COLE scenario are summarized in Figure 25.

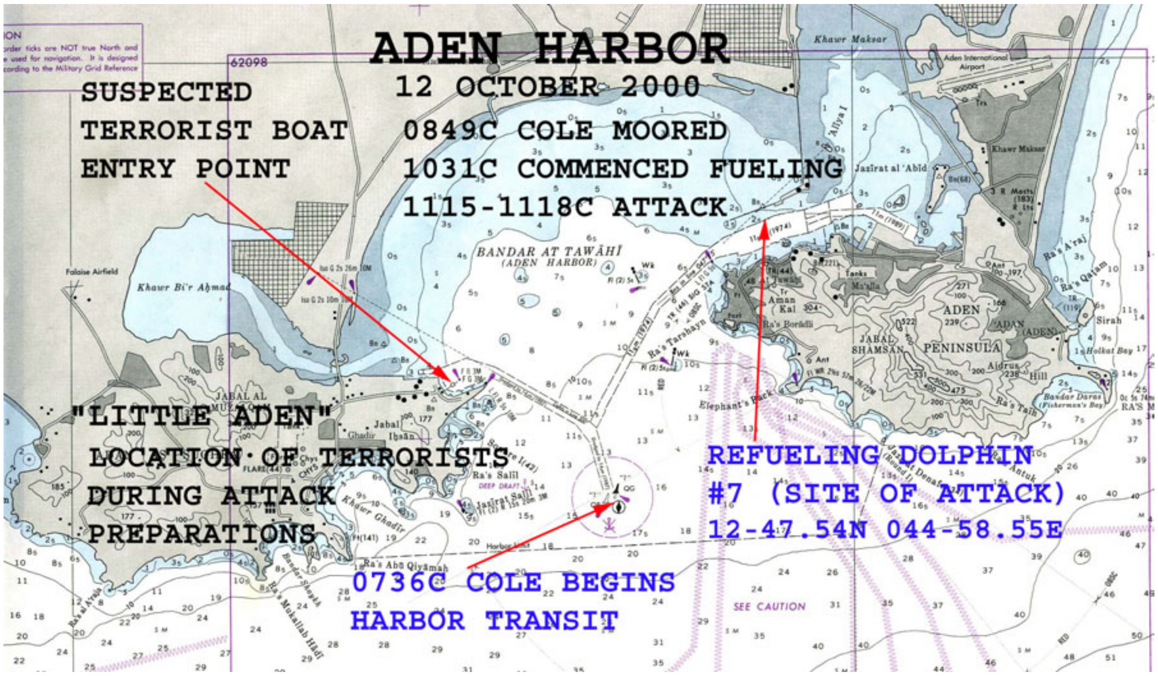


Figure 25. Nautical Chart with basic timeline of the scenario reconstruction depicted. Chart let is also utilized as the entry level view to give a snapshot to the end-user of what events are depicted as well as allowing all scenario components to load behind the scenes.

5. Dynamic Scene Playback

Once the full Cole scenario was created, the need for dynamic scenario playback came to the fore-front. Since the time to run the scenario was in excess of four hours, it became very tedious to test the scenario or demonstrate the content. As a result, the SAVAGE project development team created the Digital Virtual Display (DVD) Controller Prototype. By utilizing this controller, both a user and developer have many options in which to view and interact with a given scenario. The prototype allows one to play, pause, rewind, fast-forward, or dynamically move the scenario through the use of a slider on the bottom of the controller's 'Head's Up Display' (HUD) shown in Figure 26. Essentially this provides fine-grained visual control by the user over the X3D scene's Time Sensor functionality. Whereas before each waypoint interpolator prototype instance in a scene-graph needed a separate time-sensor or clock to drive its animation, all animation interpolators are now driven by the single untied DVD Controller instance. This authoring paradigm allows one to view multiple repetitions of critical events from

multiple viewpoints and varying levels of fidelity. Given this flexible playback capability, it is possible to analyze events from various temporal and spatial perspectives that might give greater insight than solely relying on 2D reconstructions, or unvarying 3D playback.

Figure 26. Screen capture of the Cole Reconstruction depicting the scenario being fast-forwarded to the time of attack explosion.

D.X3D CONSTRUCTION OF OTHER LOCALES

1. Naval Base, Port Hueneme

Once the necessary components for the reconstructive scenario of the attack on the USS COLE were completed, additional ports were desired to allow the end-user to dynamically create scenarios for various configurations of the scenario components. The additional ports constructed in the context of this thesis were the naval bases at Pearl Harbor, Hawaii, and Port Hueneme, California. These ports posed different problems than Aden Harbor. Primarily, Aden Harbor is distinct in the fact that it has well-defined elevation differences close to the harbor area, which resulted in a usable port construction

solely based on DTED elevation data. When trying to utilize the same technique for Port Hueneme, Figure 24 was the result.

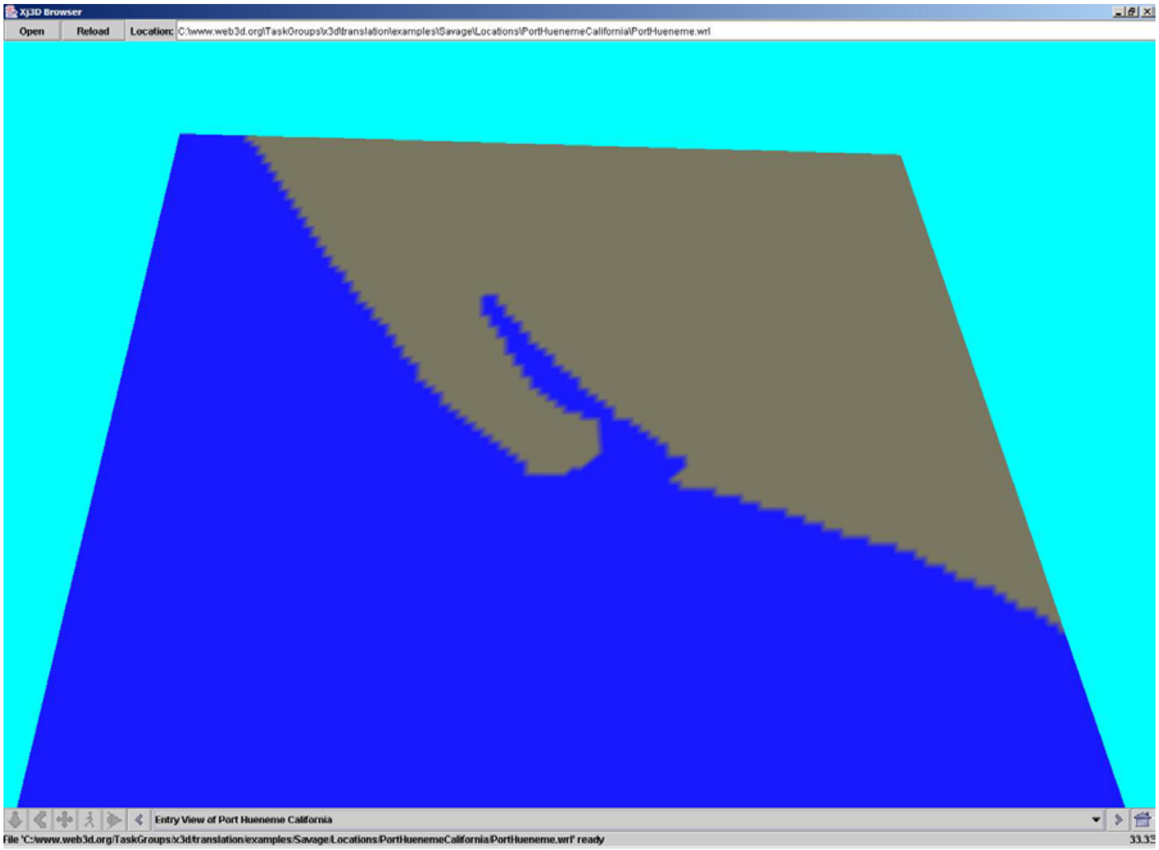


Figure27: Low resolution X3D scene of Port Hueneme, California generated from Level 1 DTED. (http://web.nps.navy.mil/~brutzman/Savage/locations/PortHuenemeCalifornia/PortHueneme.wrl) (accessed January 2003)

As can be seen, the area depicted is generally flat and leaves much to be desired. As a result, the harbor area was reconstructed based on unclassified overhead imagery and harbor charts available from the National Imagery and Mapping Area (NIMA) for this area. The resulting 3D scene is depicted in Figure 28.

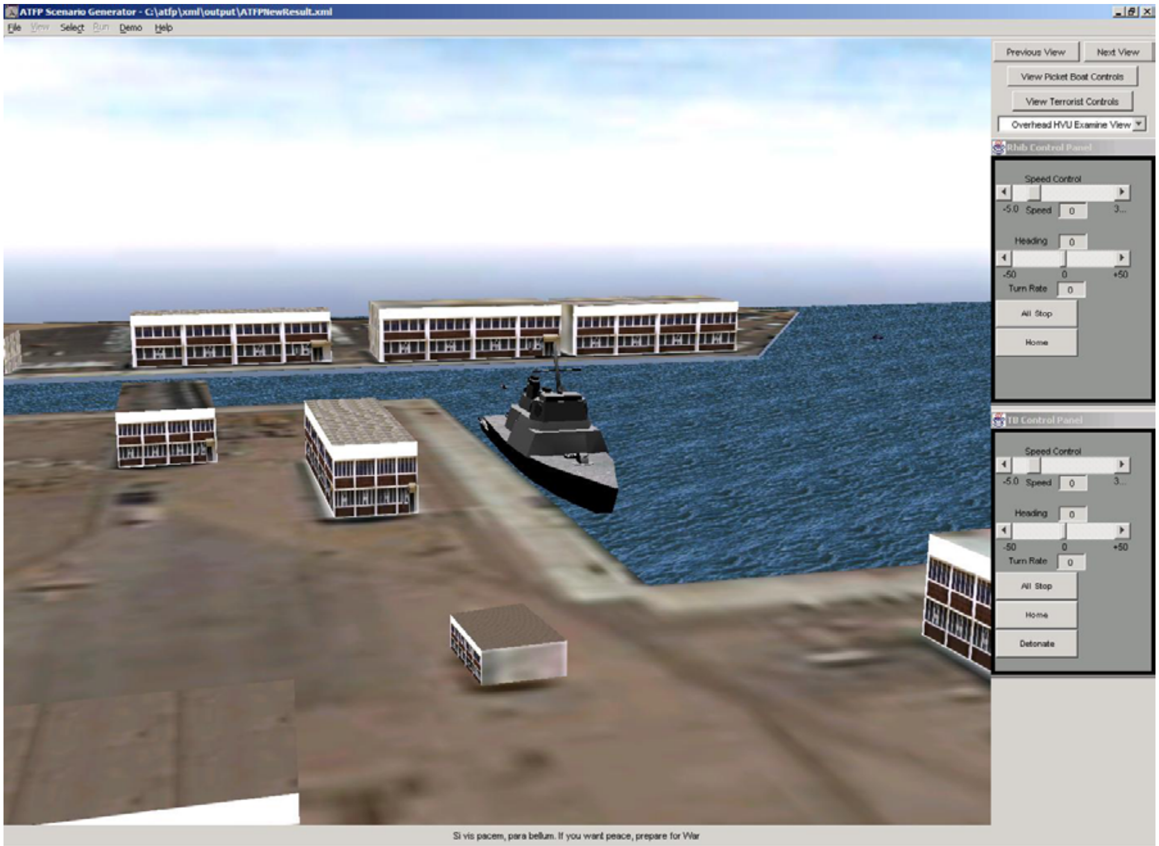


Figure28: Higher resolution scene of Port Hueneme, California depicted with a scenario in progress.

Port Hueneme is a fairly small port when compared to many other ports that Navy ships use throughout the world. No additional tools were necessary for the creation of this level of resolution port other than X3D-Edit, the nautical chart, graph paper, imagery, and textures for making more immersive appearing graphics. Modeling Pearl Harbor proved to have similar but additional challenges for the creation of a 3D scene to utilize for this work. Larger ports will likely require further tools

2. Construction of Naval Base Pearl Harbor

Pearl Harbor, Oahu Hawaii proved to have similar challenges to Port Hueneme in that the area surrounding the simulation harbor of interest was generally flat, consisting of man-made constructions. After a quick check of the SAVAGE X3D scenario database, a previous version was found that had been constructed for another project (the

USS Greenville / Ehime Maru Collision), but again the elevation data did not have sufficient detail by itself to leverage for harbor modeling (Figure 29).

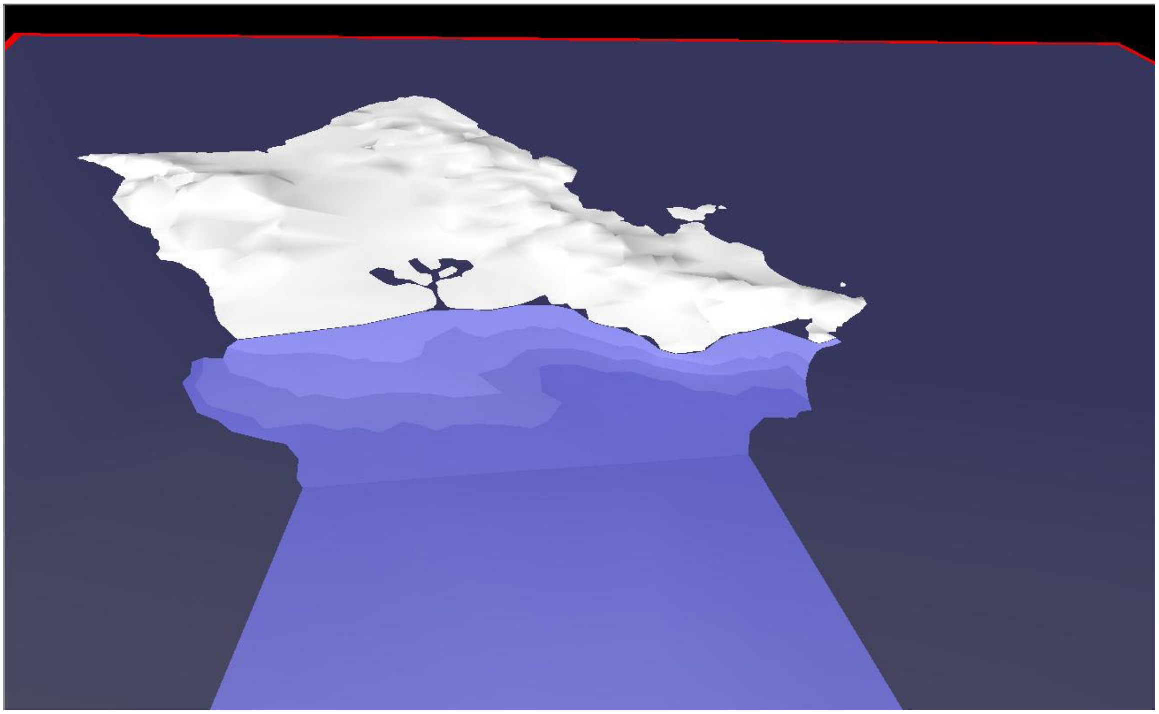


Figure29: Low Resolution X3D Scene of the island of Oahu based on Level 1 DTED and unclassified Bathymetry data. Created as part of The USS Greenville – MV Ehime Maru reconstruction (http://web.nps.navy.mil/~brutzman/Savage/Locations/Hawaii/OahuAndSouthernBathymetry.wrl) (accessed January 2003)

We then used a model of Oahu constructed by David Colleen, Planet 9 Studios, San Francisco, California, based on commercial Digital Elevation Mean (DEM) data of 10 meter resolution, and unclassified imagery acquired from previous work (Figure 30). The terrain resolution proved excellent, but nevertheless was problematic for direct use for two reasons. First, the original model from Planet 9 was not scaled to meters as the unit of measure. Default X3D/VRML units are meters, so scaling consistency is essential.

The next model of Oahu was completed in conjunction with another student thesis, based on Level 2 DTED and Mapping information utilizing the commercial modeling tools Arc View ™ and Multigen Creator. The result is a one-meter rectified

version of the island of Oahu, and then further modified by Planet 9, to contain high resolution imagery, pier outlines, and 10-meter-resolution data (Figure 31).

Figure 30. High-resolution scene of the Hawaiian island of Oahu. Courtesy of Planet9 Studios (http://web.nps.navy.mil/~brutzman/Savage/Locations/Hawaii/oahu.wrl) (accessed February 2003)

Figure31. Alternative High-resolution scene of Oahu courtesy of Major Claude Hutton, USMC. (http://web.nps.navy.mil/~brutzman/Savage/Locations/Hawaii/OahuCadrgIITSEC2002.wrl) (accessed January 2003)

The models in Figures 30 and 31 proved to be excellent examples of how one can leverage mapping or imagery information differently in order to provide the targeted client or customer of detailed terrain work the additional information needed when viewing a scene.

At this point, a higher resolution of detail was desired for Port Hueneme. As a result, leveraging an overhead image provided by NIMA of the Pearl Harbor naval base and the Multigen Creator TM modeling tool, we were able to define a fairly simplistic rapid prototyping process for ports. Figure 32 depicts the harbor scene of Pearl Harbor. Specifically critical with all of the various methods that can be utilized for creation of terrain and ports is the ability to coherently build an off-screen polygon representing the water area for the scenario and having the scene rectified to 1 3D unit being equivalent to 1 meter in the real world as recommended by the X3D specification. If the scene is rectified to meters, then the off-screen polygon correlation can be identified with some

effort on the part of the modeler, but if the scaling is not correct then the task becomes much more difficult.

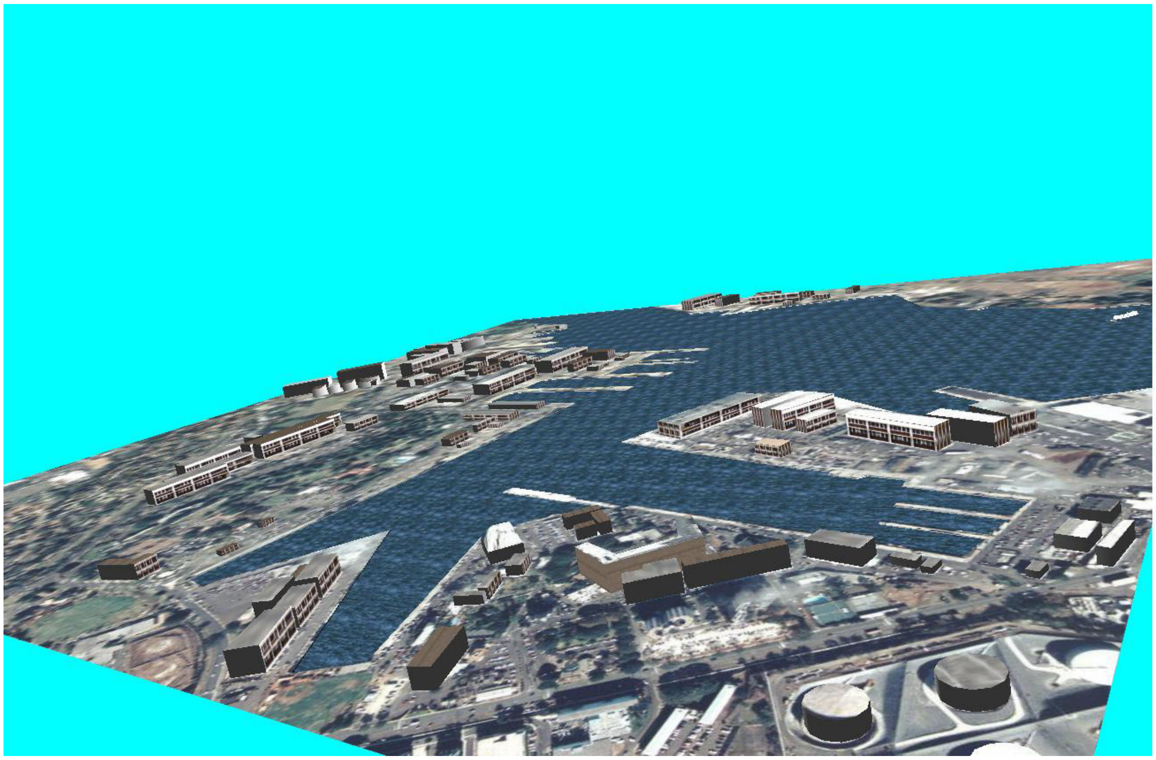


Figure 32. Result of Rapid-prototyping of the Pearl Harbor Naval base and portions of the South Channel of Oahu, Hawaii.

E. SUMMARY

This chapter has demonstrated several of the key concepts utilized when developing 3D graphics for the web, specifically through the utilization of X3D graphics. It has shown some of the basic methodologies used when generating reconstructive scenarios for dynamic replay for gaining additional insights into past or possibly future events. The building blocks introduced in this chapter are utilized in the final thesis product-- a dynamic scenario generation tool for modeling the defense of a naval surface vessel against the surface-borne terrorist threat.

THIS PAGE INTENTIONALLY LEFT BLANK

V. X3D AND BASIC PHYSICS MODELING

A. INTRODUCTION

The application of physics to real-time simulation mirrors a common problem that the computer gaming industry continually faces having a realistic look and feel to their products. [Bourg 2002] This chapter provides an overview of the basic kinematics and dynamics theory applied in this thesis, to effect changes of entity state through the utilization of the X3D Distributed Interactive Simulation (DIS) profile.

B. KINEMATICS

In [Bourg 2002], [Gomez 2000], [Lamonde 2000], and numerous other gaming, animation, and simulation references, kinematics is defined to be the study of the motion of rigid bodies or particles focused on linear and rotational position and velocity, a body and how these properties are related with respect to change over time. The effects of forces and accelerations acting on a body are considered dynamics.

For a real-time or near-real time simulation, a rigid body can be considered as an object or entity that doesn't morph or dynamically change its shape in any significant manner while moving. Location, orientation, and other factors are important, and need to be updated in real time for the targeted environment. A particle can be thought of being the center of an entity that does have mass, but the dimensions and lower levels of detail are unimportant for the problem domain being investigated. In this thesis, the primary entities of concern for simulation are ships moving on the ocean surface, in generally constrained local environments, with sea state zero.

Although one might overlook it, the units of measure for any real-time simulation system must be known and agreed upon if interoperating with others before serious work can be done. The X3D specification [X3DSPEC 2002] lists standard units of measure with respect to distance, angles, time, and color space as depicted in Table 2.

Category	Unit
Linear distance	Meters
Angles	Radians
Time	Seconds
Color space	RGB ([0.,1.], [0.,1.], [0., 1.])

Table 2. Depicts standard units of measure for X3D graphics scenes from the X3D specification.

Additionally, the default axis structure that is used in X3D graphics is followed in this thesis. That is, as the user is looking at their viewing device, the positive X axis is increasing as one moves to the right, the positive Y axis is increasing in the upwards direction, and the positive Z axis is increasing towards the viewer (Figure 33). Some 3D systems define the 'up' axis as being the Z axis as compared to the Y, such as the open source modeling and gaming toolkit, Blender 3D from http://www.blender/org (accessed February 2003).

The next property of concern after three-space position for rigid body kinematics for simulation systems is velocity. Velocity is treated as a vector with both a magnitude and direction. [Bourg 2002] This un-normalized magnitude can be thought of as the entity's speed, or rather the change in distance per a given time step in units of interest (for example miles per hour, nautical miles per hour, meters per second, etc.). Real-time animation reduces the velocity ideal to what is called instantaneous velocity, or rather the approximated velocity over a much smaller time step, usually on the order of milliseconds for real-time simulations of higher-fidelity entities.

More specifically, velocity is the amount represented in differential terms as the derivative of the change in position, or displacement (ds) with respect to time(dt) represented by: $v = ds/dt$.

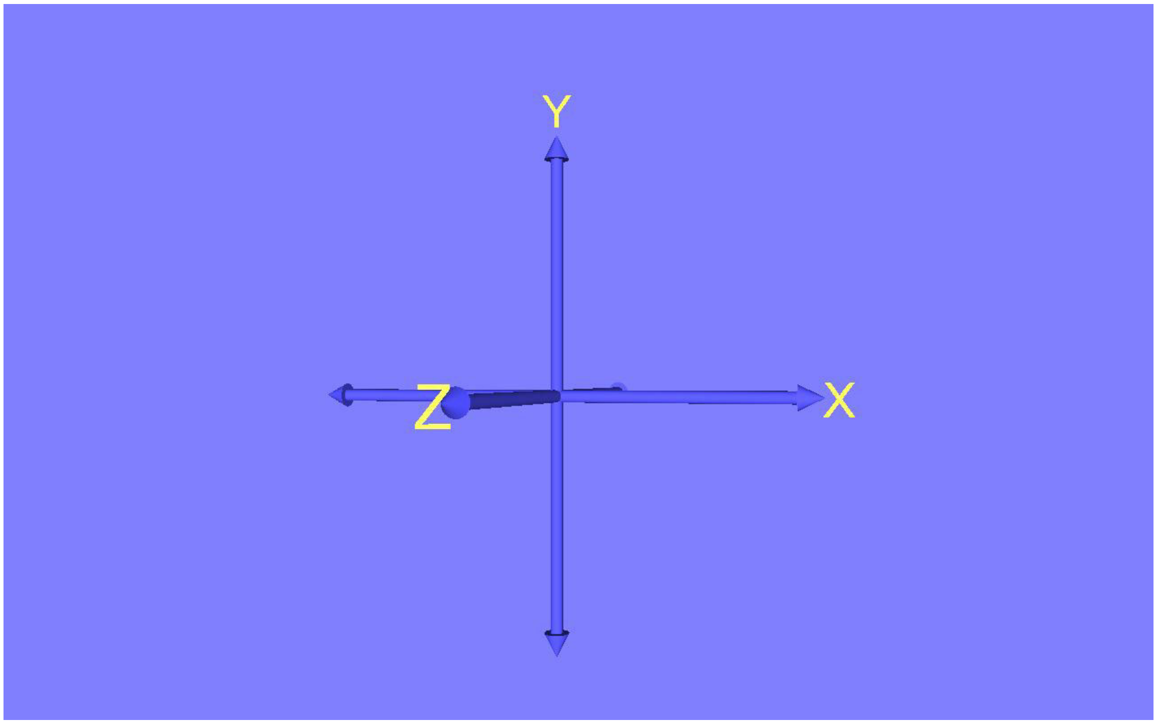


Figure 33. Depicts the default coordinate axis for VRML97 and X3D graphics scenes as viewed in the image.
(http://web.nps.navy.mil/~brutzman/Savage/tools/authoring/CoordinateAxis.wrl)
(accessed February 2003)

Now, we also care about acceleration for rigid body kinematics. If we assume instantaneous achievement of a desired speed or velocity, any insights to be gained from implementation of a simulation system could be skewed, or even worse give the experimenter false insights to system behavior. We can think of acceleration as the rate, or amount which we can increase our speed by. From [Bourg 2002], average acceleration (a) is defined to be the rate of change in velocity (dv) with respect to time (dt), represented in derivative form by: a = dv/dt.

Now, for our real-time simulation we need to produce an approximation of a given entity's velocity at a given timestep which can be dynamically determined. The Euler Method [Bourg 2002] shows that we can approximate our entity's displacement by first approximating our velocity at time T1 (where F=force, m=mass, dt=change in time or timestep).$(v_{to+dt})=v_{to} + (F/m)*dt$ Then, we can approximate the displacement at time T1(where S is the displacement).

($S_{t0+dt} = S_{t0} + dt\,(v_{t0+dt})$)

The Euler method, while useful, truncates the solution of displacement beyond the first derivative. There are well-known improvements and some alternatives to the Euler method where the percent error of actual entity properties versus the approximated ones can be significant enough to impact the stated goal or goals of a simulation system which were not investigated in the context of this thesis.

The above equations are straight forward to break into the three dimensions necessary for representation in a real time 3D graphics system by separating out the components by the three dimensions of our coordinate axis, namely the x, y, and z axis (Figure 34)

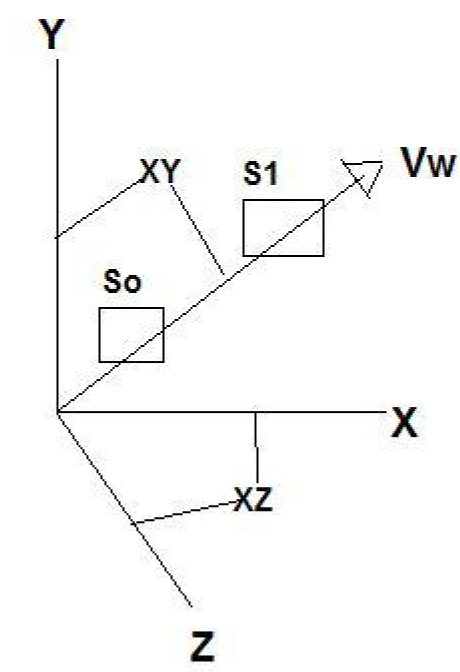


Figure 34. Depicts basic movement in 3 space by a generic entity.

So, from Figure 34, the velocity for each plane can be expressed by:

$V_x=V_w * \cos(\text{Theta}_x)$

$V_y=V_w * \cos(\text{Theta}_y)$

$V_z=V_w * \cos(\text{Theta}_z)$,

Where (Theta_x) is equal to the displacement traveled in the X plane divided by the overall displacement; similarly for the other two planes. The 3D displacement can be calculated by multiplying the value for each component's respective velocity by the time. Acceleration is also incorporated in the same manner discussed earlier for one dimension, we just split into three axes as we did for the velocities.

Angular displacement, velocity, and acceleration are similarly calculated as their positional counterparts. For average angular acceleration calculations, the angular displacement at T_1 is equal to the angular displacement at T_0 + the angular velocity at T_1 * the time step + the angular acceleration. Further complicity and fidelity can be gained by additional calculations with respect the effects upon a rigid body's angular acceleration by both tangential and centripetal accelerations.

The desired end-state for our system is to have ships move in a kinematics based fashion from ordered rudder angles and speeds initiated either by an end-user or autonomous agent (Figure 36). For each timestep in our real time simulation, the actions in figure 35 are carried out and the entity's state updated.

For Each Time Step

1 - Update the acceleration vectors in each plane
2 - Update the velocity vectors in each plane
3 - Update the entity's roll, pitch, and yaw angles
4 - Update the entity's x,y,z position

Figure 35. High level view of the kinematics physics state update for entities in the AT/FP Scenario System.

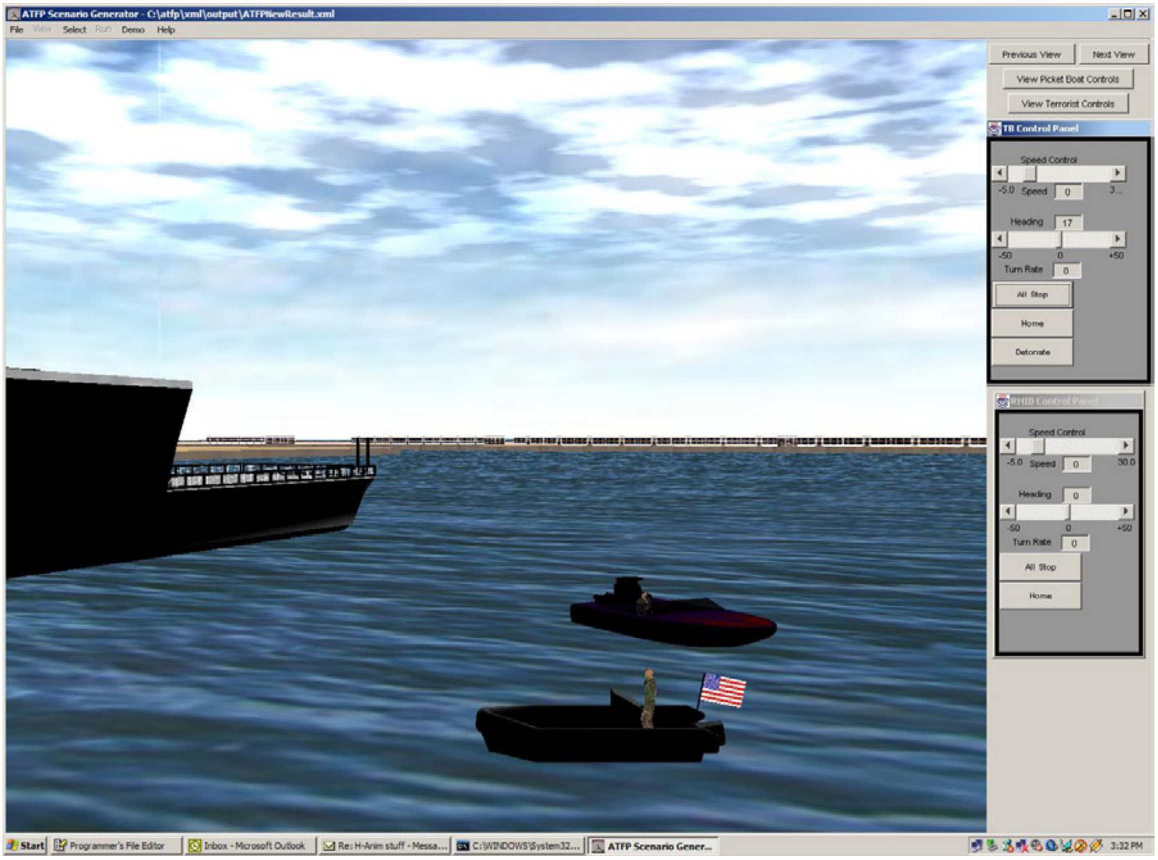


Figure 36. Real-time scenario in progress. Two small boats and one Arleigh Burke class destroyer depicted being controlled by kinematics-based physics controllers.

C. DIS-JAVA-VRML

It was decided in the design phase of this thesis to leverage a socket-based simulation system to maintain a separation between the system's models and controllers from the various graphical views that could possibly be associated with them in order to minimize the impact that changes in rendering structure from private industry would have on the project. After choosing a socket-based infrastructure, it was further decided to utilize the DIS protocol. DIS is an ISO-approved standard network protocol, so as a result it is well-defined and proven to be scalable. Second, much work has been completed in the past proving DIS-driven simulations can be run in lightweight X3D/VRML97 clients in web browsers as well as in more heavyweight client side applications. The primary packet type of DIS leveraged was the Entity State Protocol Data Unit (ESPDU), which communicates all of the information needed in order to update the physics state variables mentioned above. Additionally, DIS also supports

other protocols important to our simulation, such as the Detonation Protocol Data Unit and Firing Protocol Data Unit that allow us to communicate state information regarding the expenditure of weapons, and if an entity has been hit by a weapon. With the advent of the X3D specification, the DIS profile is a full-fledged member profile of the proposed specification which should guarantee continued support throughout the near future. The only difference made in the physics modeling steps from Figure 35 is that at the end of each decision loop for each entity at each time step the entity transmits an ESPDU update to the applicable networking Internet Protocol (IP) address.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. INTERFACE DESIGN AND IMPLEMENTATION PROCESS

A. INTRODUCTION

At the onset of development of work on this thesis, the decision was made to focus specific attention towards maintaining a 'user-centric' design and implementation process in order to best provide an exemplar of how current web-based modeling and simulation technologies can be leveraged towards aiding the warfighter in gaining insights on the employment of tactical doctrine. Traditionally, usability is a neglected (or poorly implemented) area with respect to Department of Defense (DoD) Information Technology (IT) applications. As a result, many systems are considered technically successful and delivered to the fleet, but to the end-user have many short comings such as: being very confusing to operate, require many days away from one's job to attend a special school or training course in order to learn how to use the application, are error prone, hard to leverage to their full design capabilities. [Darken 2002] This thesis leveraged proven industry practices in the field of usability [Darken 2002][Nielsen 2000] in order to reduce the sum total number of errors for the application produced, decrease any required training time for use in the training and education environment of the application, attract other interested parties in the Department of Defense for collaboration on research, and just make it more intuitive for warfighter's to utilize.

B. NEEDS ANALYSIS

The goal of this work is to provide a web-based simulation to the U.S. Navy fleet in support of gaining insight for Anti-Terrorism/Force-Protection (AT/FP) doctrinal requirements at the individual ship level. Currently, there is no organic capability supporting the modeling and simulation of AT/FP defenses on most U.S. Navy warships that might allow Force Protection Officers and their support personnel, to compare and contrast potential employments of the limited available resources onboard ship, and to view potential outcomes in the numerous ports and harbors that warships visit throughout the year.

One additional area currently under doctrinal development by the Navy is the formulation of a Detect-Sort-Decide-Engage (DSDE) decision process to aid servicemen in making correct and timely decisions in defense of the ship by providing them with pre-assigned distances to identify unknown personnel and vehicles (air, land, and sea), and to employ various lethal and non-lethal actions based on implemented rules of engagement. Until official doctrine is thoroughly tested and evaluated, ship's force personnel will need to be able to view the effect of varying the ranges to perform different actions in the DSDE decision loop, based on experience regarding the location of the ship. This is a very difficult job due to dramatic differences in harbor layouts throughout the world. Thus, this thesis application might have significant impact if it can be effectively used by naval operators

1. Judgment Criteria

The criteria that will be used to judge the effectiveness of the application's performance are:

a. (learnability) Ease of use. Minimal to no training required to use, meaning that it should be intuitively obvious to set up and test defenses without formal training in use of the software.

b. Scope for location availability is limited (i.e. only enough ports are implemented in order to adequately perform testing and development). Efficiency is judged based on different positions of tactical entities in the available harbors only.

c. Formatting of results of the simulation need to be user friendly and meaningful.

d. Feasibility of use as a practical planning tool as opposed to traditional 2D pen and paper methods. The tool needs to be no harder than these traditional methods to utilize and require (at worst) equal time to complete planning a defense.

e. Predictable error behavior and minimal error rate. Provide ability for the user to go back to the previous state to avoid re-work when user makes an error.

f. Memorability. Built-in demo / teaching mode since long terms between uses can occur.

2. Problem Approach

The developmental focus done in conjunction with the usability design and analysis focuses on:

a. Portability: (considered pivotal to deployment onboard navy ships). Multiple platforms and operating systems targeted for possible deployment. Aiming for as close to a 'one-click' installation process as possible.

b. Graphical User Interface (GUI): Use current windows-type interface. Change or redesign to meet the expectations of the above judgment criteria.

c. Add additional functionality not yet existing and maintain focus on judging criteria while developing the remaining portions of the interface.

d. Develop useful reports and printable output describing the results of a simulation run.

C. PROJECT GOAL

The primary goal for this project was to enable the user to virtually place a ship in a computer simulated harbor for the purpose of gaining insight into anticipated effectiveness of specific Anti-Terrorism/Force Protection (AT/FP) defensive postures. Features that help to meet this goal consist of but are not limited to:

1. Easily choosing a harbor for a simulation run.
2. Easily choosing a ship type.
3. Easily choosing a placement of the ship type within the chosen harbor.
4. Easily choosing placement of the required defenses. In this case limited to contain the surface-borne terrorist threat only with extensions to other threats at a later time frame.
5. Easily choosing appropriate ranges for the defensive entities to act in the Detect-Sort-Decide-Engage process.

6. Easy configuration of a current threat condition to identify applicable rules of engagement.
7. Easy configuration of threats to run simulated attacks against the configured defensive parameters.
8. Intuitive viewing and interaction with the simulation
9. Easy ability to print the 2D view of the scenario setup and configuration for further annotation/markup, training and review.
10. Easily able to save settings and simulation runs as desired and required for subsequent restart/replay.

D. USER ANALYSIS

Targeted user groups for this application are U.S. Navy Surface Warfare officers plus support personnel such as Fire Controlmen and Gunner's Mates that plan Force Protection and Anti-Terrorism (AT/FP) defenses for surface ships or submarines. They are experts in their fields and are formally trained in AT/FP tactics and methodologies by the navy. They are competent in the use of computer programs in windowing environments such as Microsoft Word, Outlook, and both the Internet Explorer and Netscape web browsers, and are guaranteed to have access to modern computers with the above mentioned software loaded and operational. They have access to Information Technology (IT) trained professionals in their work environment for troubleshooting and assistance with computer hardware, software, and any other technical issues that may arise.

1. User Characteristics

This application will be used to plan shipboard AT/FP defenses, and identify possible shortcomings in defensive plans. Rate of usage is expected to be bi-weekly overseas, and at least quarterly in the continental United States.

2. User Skill Levels

With the widespread implementation of IT-21 throughout the U.S. Naval Surface Force we can assume our targeted user base has general computing skills with some

proficiency with Microsoft Office Products in Windows Operating System environments, web-surfing capable, and experience utilizing non-user centric designed software for years. Also, we can assume marginal to good typing skills, but this is not a requirement for this application.

3. Conclusion

Threats and capabilities in the future are expected to change, the application design needs to take this into consideration, and plan for the capability to add items at future dates.

E. TASK ANALYSIS

The task analysis identifies all required tasks the user can be expected to perform while utilizing the application and serves as the basis by which we will later test and record measures of effectiveness assessing the application's successful usage.

<u>Primary Task 1. Setup scenario</u>:

Subtasks:

1a) Choose harbor for simulation run.

1b) Choose ship type

1c) Choose placement of ship in harbor.

1d) Choose placement of required defenses about ship in harbor.

1e) Choose appropriate ranges for the defensive entities to act in the defensive process.

1f) Configure applicable defense model parameters for the scenario run.

1g) Choose and configure threats(s) to run against the configured defense parameters.

1h) Choose and configure user participation mode. (ie agent only, user only, or a combination of each).

1i) Print 2D layout of defensive configurations.

1j) Save scenario setup to disk for future reuse.

Primary Task 2. Run Simulation:

Subtasks:

2a) Observe 3D visualization of scenario situation

2b) Choose current viewpoint and be able to change to other viewpoints in the 3D scenario run.

2c) Show capability of being able to view agent and/or user controls for entities in the scenario.

2d) Show capability to view end of run analysis after scenario run is complete.

2e) Demonstrate capability to load and run scenario from scenario saved to disk.

2f) Demonstrate capability to run scenario for statistics after viewing a single scenario run.

F. CONCEPTUAL DESIGN

Before starting to visually design and implement the user interface for the application, a conceptual design for the requirements of the objects, attributes on these objects, and the exposed relationships and actions for the system was conducted. Results are presented in Figure 37.

Objects	Actions On Object	Attributes	Actions on Attributes	Relationships	Actions on Relationships
Scenario	New Open Save SaveAs Run RunAndRecord RunAndOutputToFile	NewOrPreexisting	setAsNew SetAsPre-existing	1 Scenario has 1 Location 1 Scenario has 1 Ship 1 Scenario has 1 NavTrack 1 Scenario has 1 or more Terrorist	choose the location chose and place the ship choose the navtrack choose and place the terrorist
Location	Load Close Preview	Name 2D 3D	getName get2D get3D		
Ship	Load Close	Class/Type length width height ownship Friend/Foe/Neutral		1 Ship deploys 1 or more Small Boat 1 Ship has 1 or more Sentry 1 Ship has 1 Armory	DeploySmallBoat RecoverSmallBoat AddSentry RemoveSentry
NavTrack	new open save saveAs	waypoints mooringLocation mooredHeading SOA	SetWaypoints setMooringLocation setMooringHeading setSOA		
Small Boat	edit save	picketStartPoint weapon	setPicketStartPoint setWeapon	I Small Boat has 1 Sentry	AddSentry RemoveSentry
Sentry	new open save saveAs delete	ID Weapon Decision Ring Ranges Location	setID setWeapon setRanges setLocation	1 sentry has 1 weapon	DrawWeaponFromArmory ReturnWeaponToArmory
Decision Ring	setTypeOfRing setRange setCenterOn	4Rings	SetRanges		
Terroist	setExpectedThreatLevel setWeapon	ExpectedThreatLevel Weapon			
Armory	getWeapon addWeapon removeWeapon	ListOfWeapons	getListOfWeapons	1 Armory has 1 or more Weapons	add/remove weapons from inventory
Weapons		Type range	getType getRange		

Figure 37. Conceptual Design of the AT/FP Scenario Generator interface requirements completed prior to rapid prototyping efforts.

G. VISUAL DESIGN

Prior to implementation of a prototype version of the user-interface (UI) of the AT/FP scenario generation application, a hand-drawn version of the application UI was completed and tested on one novice-level and one expert-level test subject, in order to

gain insights towards design flaws beforehand. Specific comments from the test subjects were:

Reviewer One Comments: (from an 'expert' potential user of the proposed system): "The GUI for the AT/FP Simulator looks like it should be very user-friendly, especially with the implementation of the Wizard setup. I just have two questions:

1. Under Panel #4 of the wizard setup, what are the S, D, A, and E boxes used for?

2. For the same panel, how will you implement the feature for the user to select a type of defender (M-14, shotgun, etc.) and place him at a specific location on the ship? Will it be a select defender/drag-and-drop him to a location or something else?

Reviewer Two Comments: (from a 'novice' user):

1. Would like dialog boxes explaining the step that the wizard is on so you don't have to guess.

2. Thought having text input for ranges would be a good alternative to dragging and dropping those as well.

3. Thought a demo mode would be a good complement to the wizard mode."

The major portions of the hand-drawn UI prototype are shown in Figures 38, 39, 40, and 41.

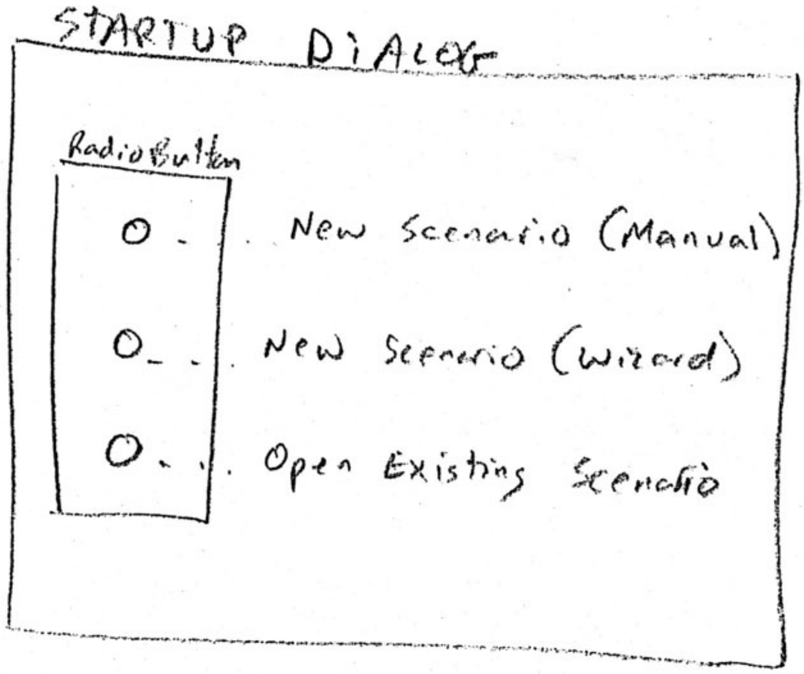


Figure 38. Conceptual picture of the startup dialog completed as part of the rapid visual design for the application.

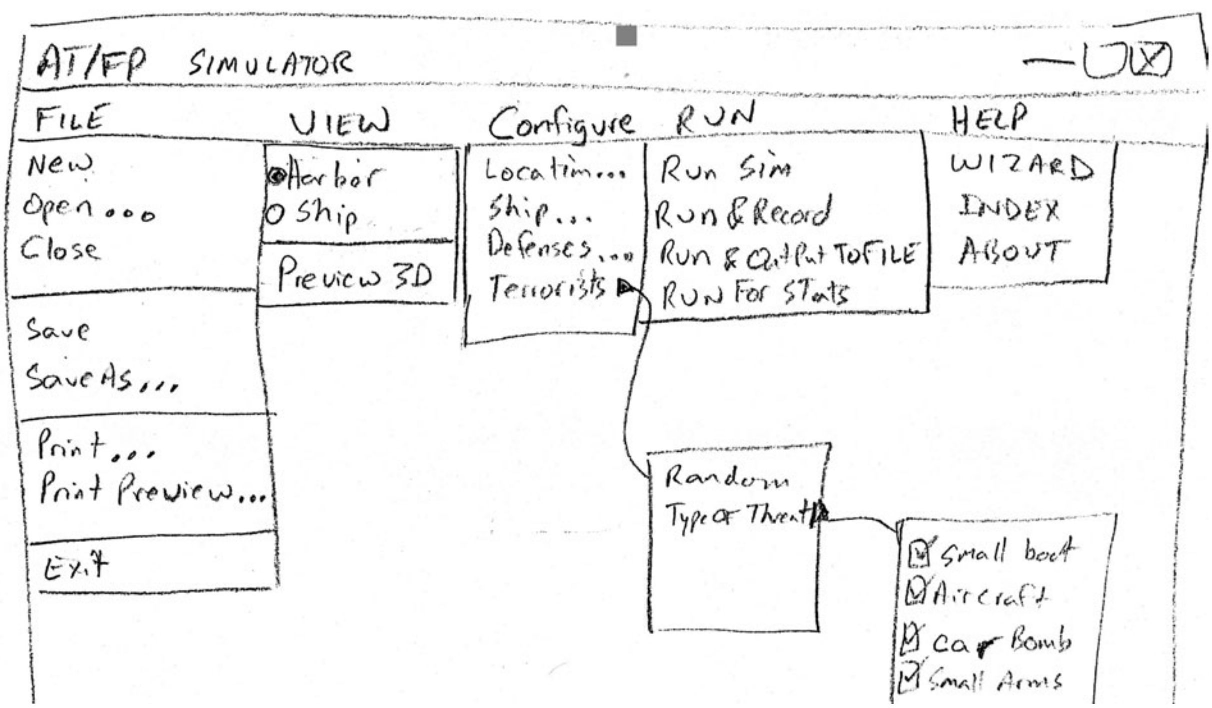


Figure 39. Main User Interface (UI) menu with options from the rapid visual design.

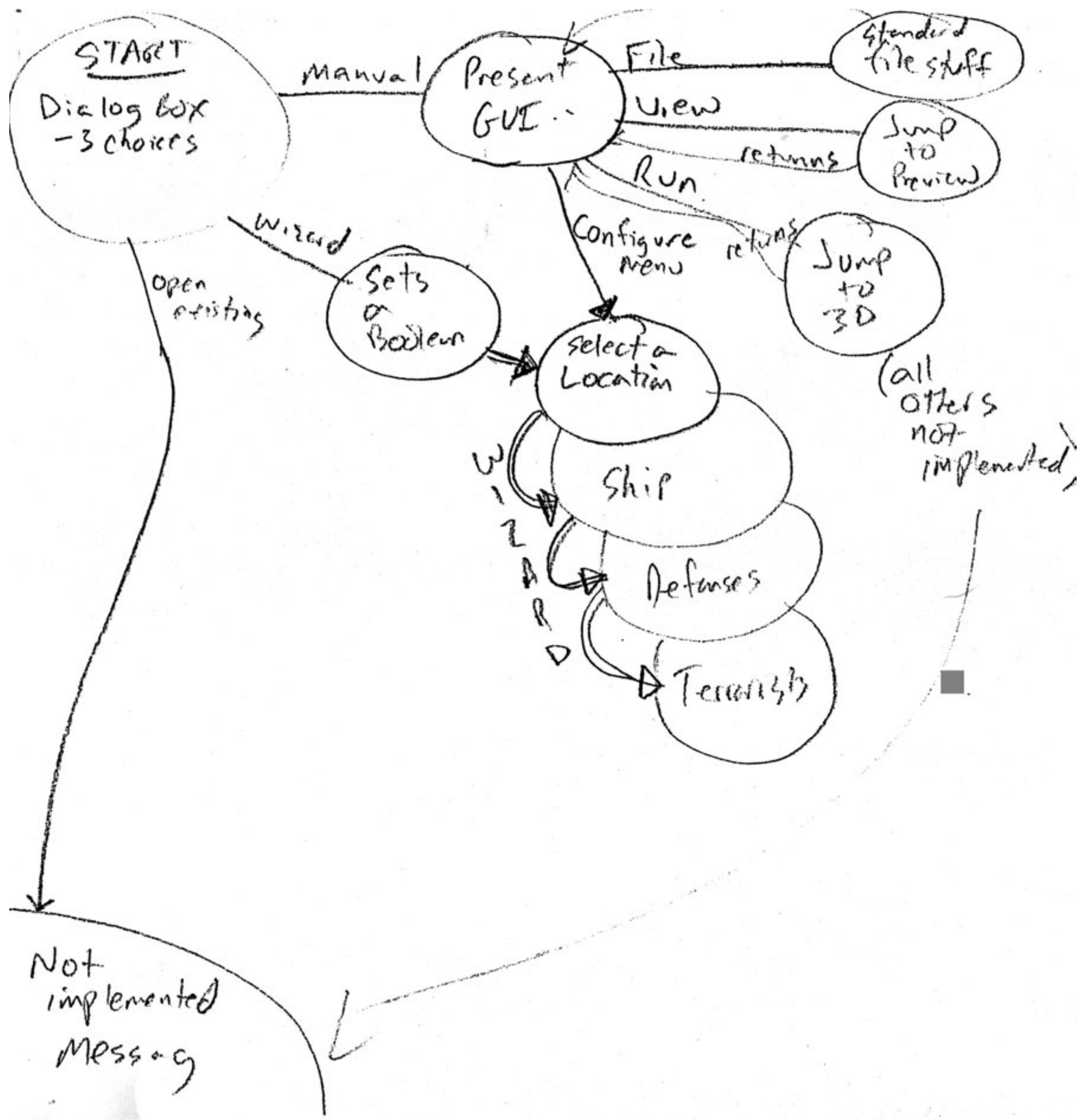


Figure 40. Depicts available options from the application User Interface (UI).

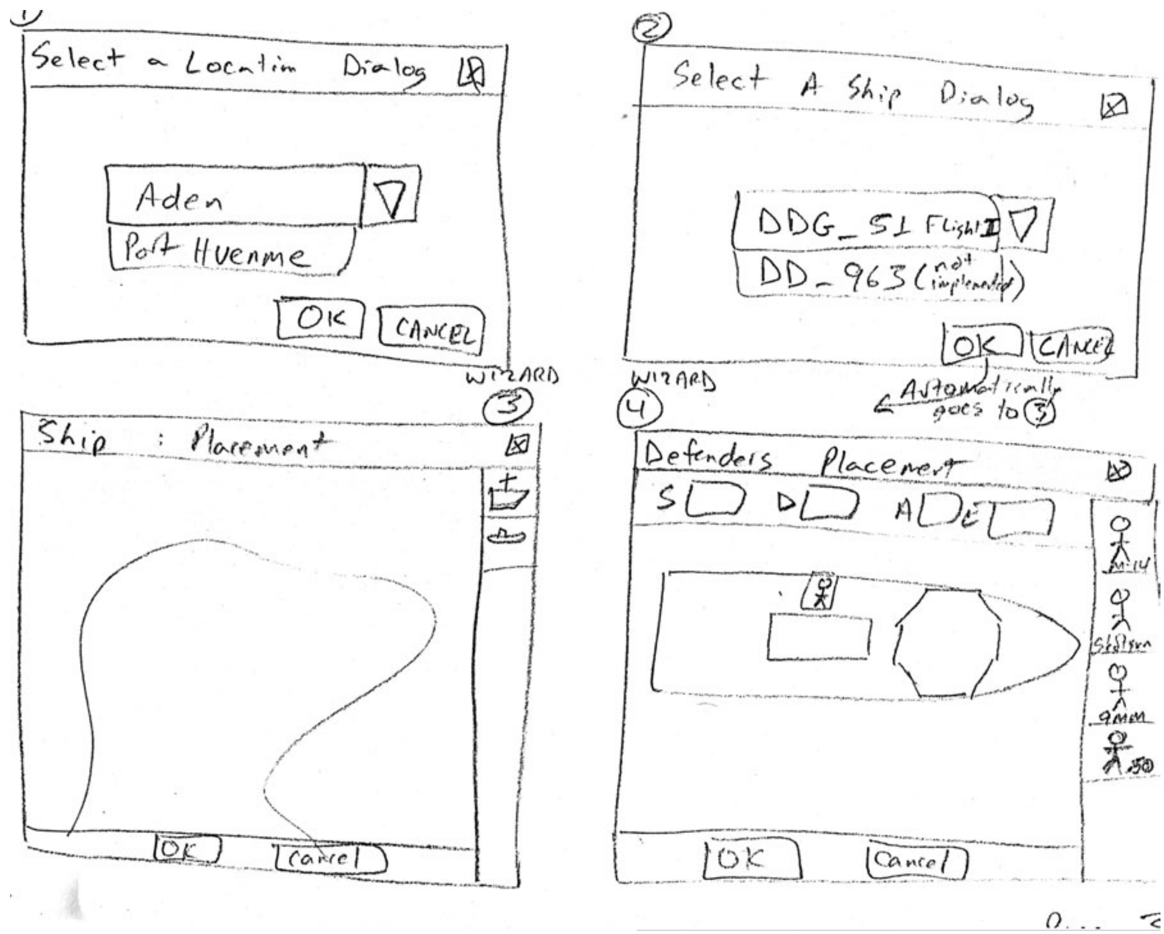


Figure 41. Depicts the basic actions prototyped for the end-user to take while configuring the defensive setup for an AT/FP scenario.

It became important to conduct several design iterations on paper before starting the actual development of the interface. From the conceptual model design to implementation, we identified many places where development time would have been wasted on implementation, correction, implementation, etc. if time had not been taken to discuss alternatives in a group setting over a period of several days. Some ideas that made sense early in the design discussions evolved significantly in other directions by the time subsequent 'pen and paper' analysis was conducted with two test subjects.

H. USABILITY ANALYSIS

The following tasks were carried over from the earlier task analysis for the initial usability assessment conducted on the AT/FP Scenario Generator application. These tasks constitute the specific areas of interest for the usability study.

1. Setup a simulation
    a. Choose a harbor
        i. Menu selection
    b. Choose a ship
        i. Menu selection
    c. Choose placement in the harbor
        i. Direct manipulation
    d. Choose placement of required defenses
        i. Direct manipulation
    e. Choose appropriate ranges.
        i. Form fill in
    f. Choose threat.
        i. Menu selection
    g. Choose participation mode.
        i. Menu selection
    h. Print 2d layout.
        i. Menu selection
    i. Save for future use.
        i. Menu selection
2. Run a simulation.

a. Observe 3D scenario.

b. Choose viewpoint.

    i. Virtual environment controls

c. Choose speed.

    i. Virtual environment controls

d. Stop and start simulation.

    i. Virtual environment controls/Menu Selection

e. Restart simulation.

    i. Virtual environment controls

f. Record simulation for playback later.

    i. Virtual environment controls/Menu selection

I. USABILITY TEST SUBJECTS

In total, seven test subjects were subsequently tested for the initial usability testing phase for the AT/FP Scenario Generator application. Six of the test subjects were active duty naval officers, and one was a prospective naval officer currently in his last year of Reserve Officer Training. Although by no means all-encompassing, this number of test subjects was considered adequate for an initial round of testing.

Several additional assumptions can be made regarding the targeted user base for this work. First, the targeted end-users were U.S. Navy Surface Warfare Officers, other sea-going naval officers, and support personnel such as fire controlmen and gunner's mates that plan AT/FP defenses for navy warships and submarines. They are experts in their fields and are formally trained in AT/FP tactics, techniques, and procedures by the navy. They are assumed to be competent in the use of computer programs in windowing

environments such as the Microsoft Office suite as well as capable of web-surfing in the Netscape and Internet Explorer web browser applications. They are also guaranteed to have access to current computer hardware with the above-mentioned software deployed and operational. They also have access to IT trained professionals in their work environment for troubleshooting and assistance with computer hardware, software, and other technical issues that may arise.

J. DATA COLLECTION AND JUDGEMENT CRITERIA

Measure Of Effectiveness 1. Ease of Use: (Learnability) Minimal to no training required to use. Should be intuitively obvious to set up and test defenses without formal training in the use of the software.

Some interesting results were found in this area. During the design phase, it was decided to have two forms of scenario editing/creation/modification available to the end-user: 1) Manual data entry and 2) a wizard assisted mode. Testing found that there were minimal errors and training required while using the wizard mode compared to the manual entry, yet over 50% of the tested users still preferred the manual entry although their performance was sometimes much worse. Testing also revealed that even though the wizard mode was helpful in eliminating errors and decreasing editing time among the end-users, utilizing only text-based information in describing what actions to take next did not seem to fully work in all cases. Either too much text appeared in the dialog, or else there seemed to be a need to incorporate descriptive pictures as well, especially for 3D scene manipulation. None of the tested user-base had prior experience with 3D scene manipulation in a web browser. No user was able to adequately manipulate the scenes with a text-based description before loading from the wizard without a fair amount of trial and error with the interface. Thus a training movie and demonstrated tutorial appears to be very important. After this period, the majority of tested users enjoyed and found the 3D worthwhile.

Moved to next page

The first result examined with respect to learnability is that of overall time to complete a task. Participants were given a list of tasks to perform, the start and finish times were noted to determine overall time for completion of the tasks. The tasks on the first three lists were exactly the same. Participants were prompted to use the manual mode for the first list, the wizard mode for the second list, and then allowed to make their own decision for the third. Results are shown in Figure 42.

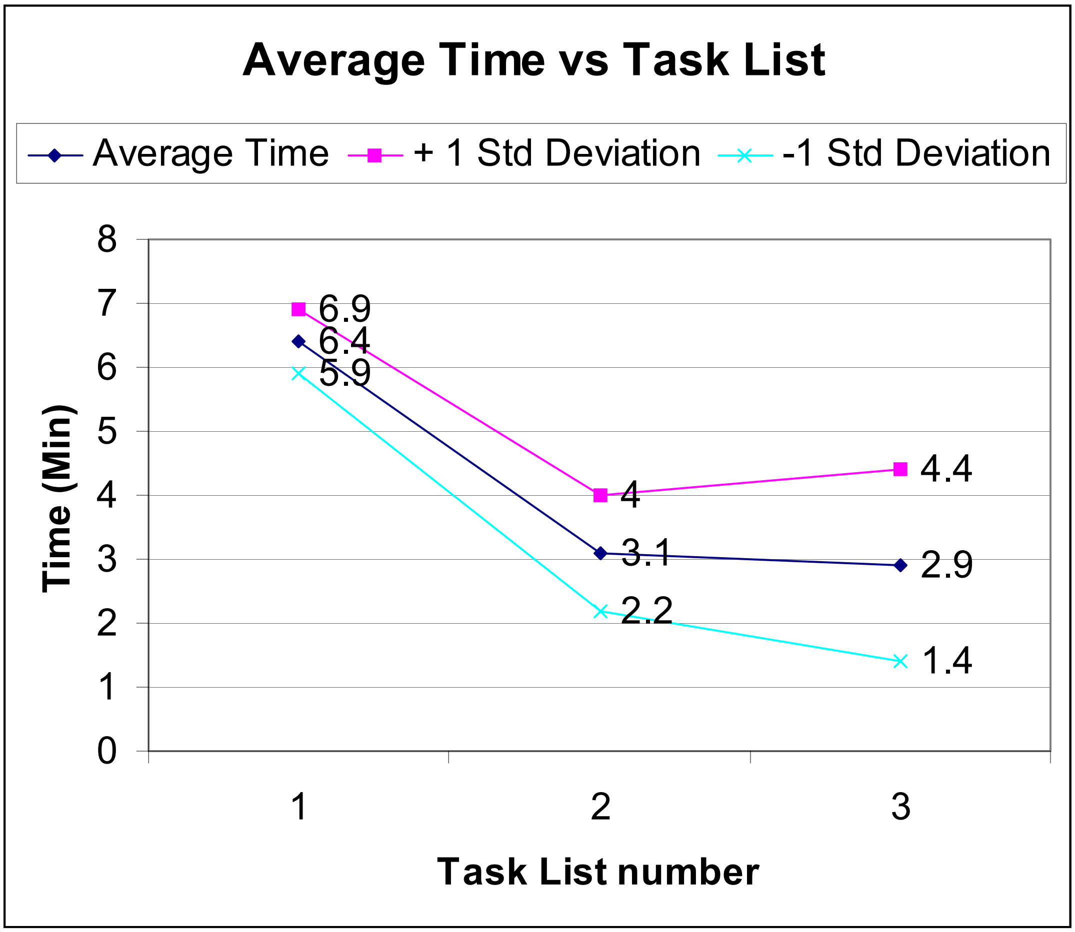


Figure 42. Depicts the average time versus task list from the initial AT/FP Scenario Generator Usability Study.

All participants were able to complete the task list faster with experience. Interestingly, average time for list 3 is less than for list 2, even though most participants decided not to use the wizard when given the choice. Testing also counted the number of good clicks, bad clicks, and keyboard/mouse switches. Evaluation considered a good

click to be anything that is in the correct direction of completing the current task item. For instance, clicking on the right menu or using the ALT+<key> combination to open the correct menu for the current task is considered ‘good’. Bad clicks were anything that indicated that the user did not know how to proceed. Clicking on the wrong menu, for example, is counter to completing the task, and so we considered it an error. Figure 43 chart shows the results of errors averaged across all the participants.

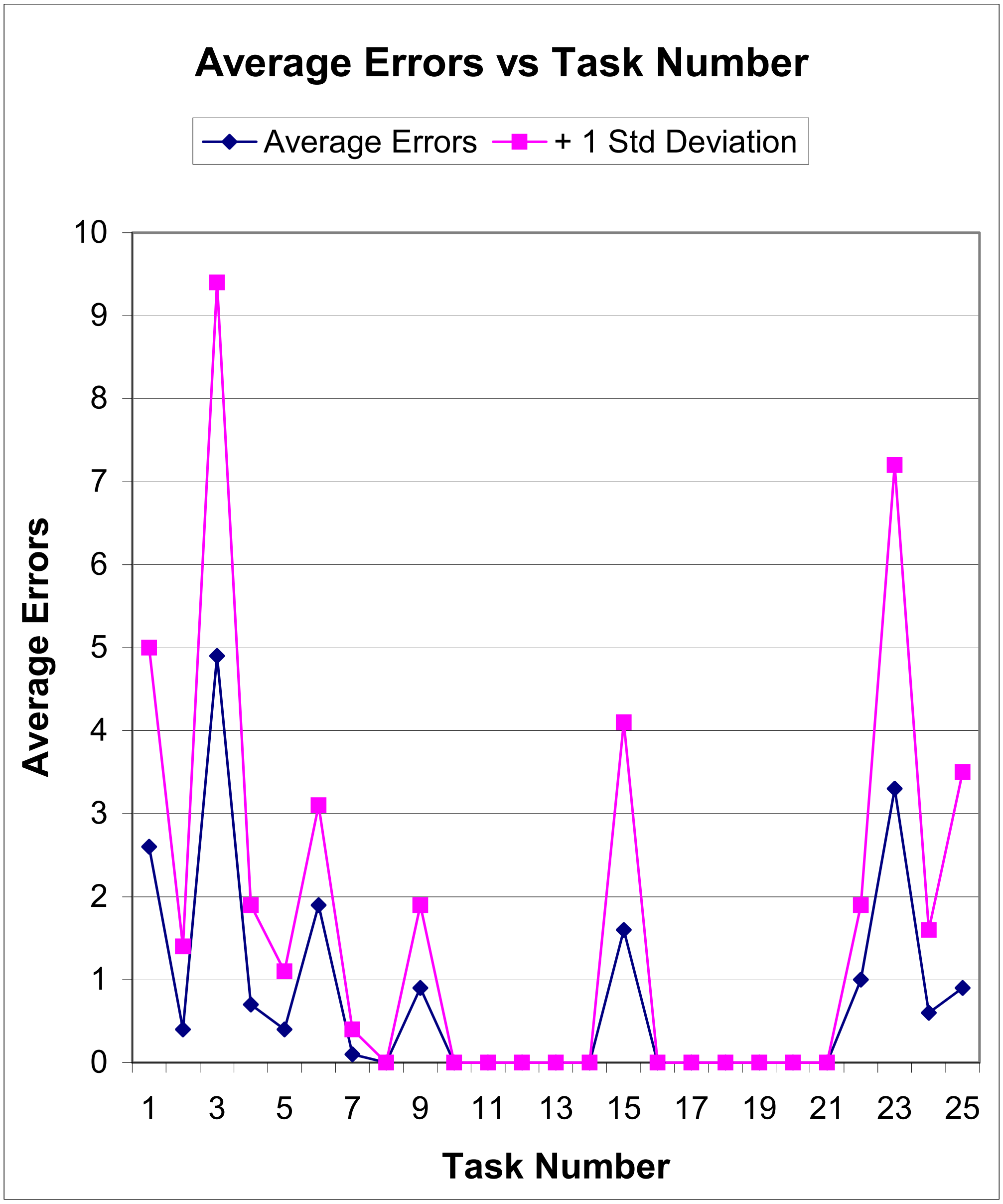


Figure 43. Depicts the Average Number of Errors versus the Task Number

The tasks 1 to 6 in Figure 43 correspond to the first task list. Tasks 7 to 12 correspond to task list number two. Tasks 13 to 18 correspond to the errors for task list number three, and the remaining task numbers are from the miscellaneous task listing. Notice that the error rate for tasks 13 to 18 is almost zero, except for task 15. Task 15 was the direct manipulation task for placing the ship and small boat. Besides that task, the users had learned the interface well enough by the third iteration to significantly reduce error rate. This data considered with the average time data shown above indicates that the application is learnable.

When participants finished specific task lists, they were presented with a questionnaire to capture their thoughts about the tasks they had just completed on a scale of 1 to 5 where an answer of 1 means that the task was consider easy to accomplish and 5 being hard to accomplish. These results are shown in the next series of charts. Each of the seven participants (shown along the horizontal independent axis) was asked three ease-of-use questions. The bars for each user show their answers to the questions with a longer bar indicating greater difficulty. This first chart (Figure 44)shows that most users found the application moderately difficult to use the first time, but there are no ‘too hard’ reactions, which was answer number 5 in the questionnaire.

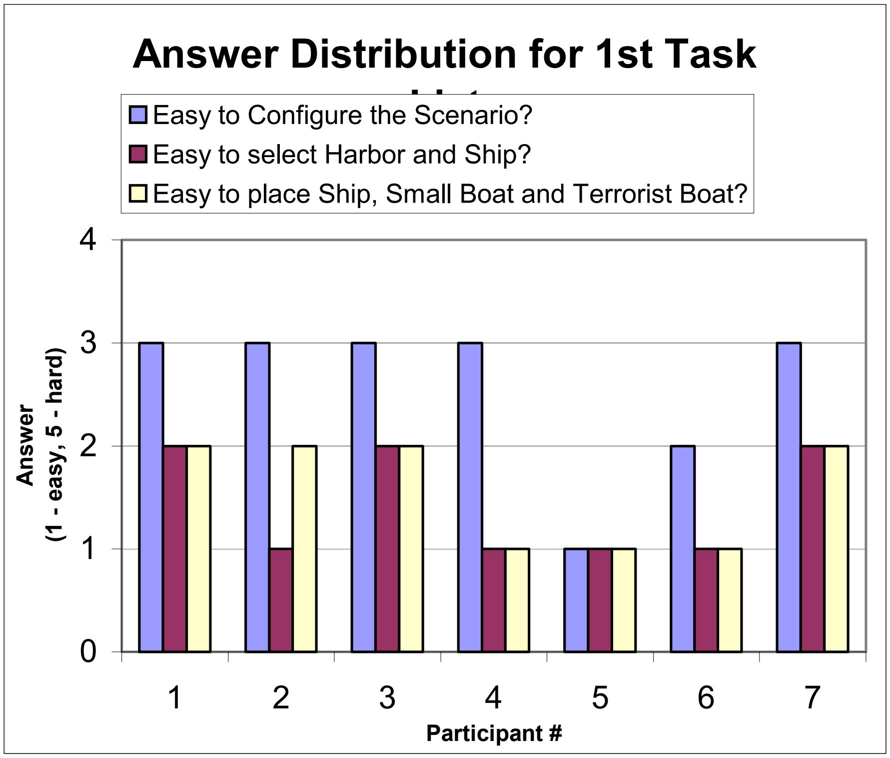


Figure 44. Depicts the answer distribution for the 1st task difficulty assessment in manual mode.

Recall that task list two was completed by using a wizard interface that prompted the user with dialog boxes for every step of the task list. All users found the interaction when guided by the wizard to be obvious (Figure 45).

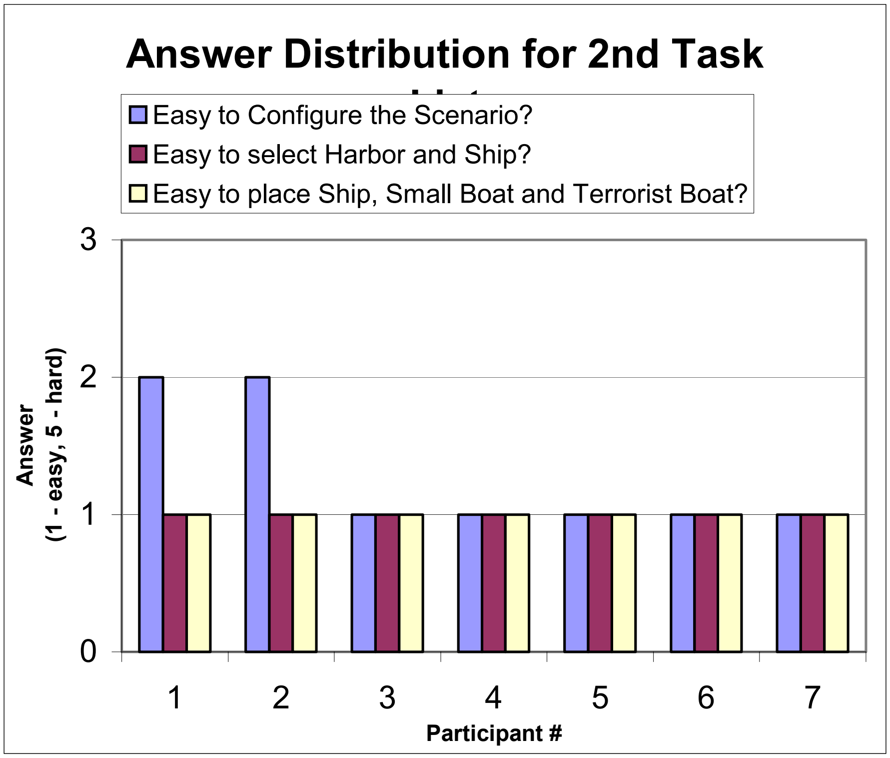


Figure 45. Depicts the answer distribution for the 2nd Task. Level of difficulty assessment for Wizard mode.

After the third time completing the task list, questions were asked about how the participant felt about the ease-of-use of the application again. Results show some moderately difficult answers appearing again (Figure 46). Remember that many participants opted not to use the wizard for the third task list.

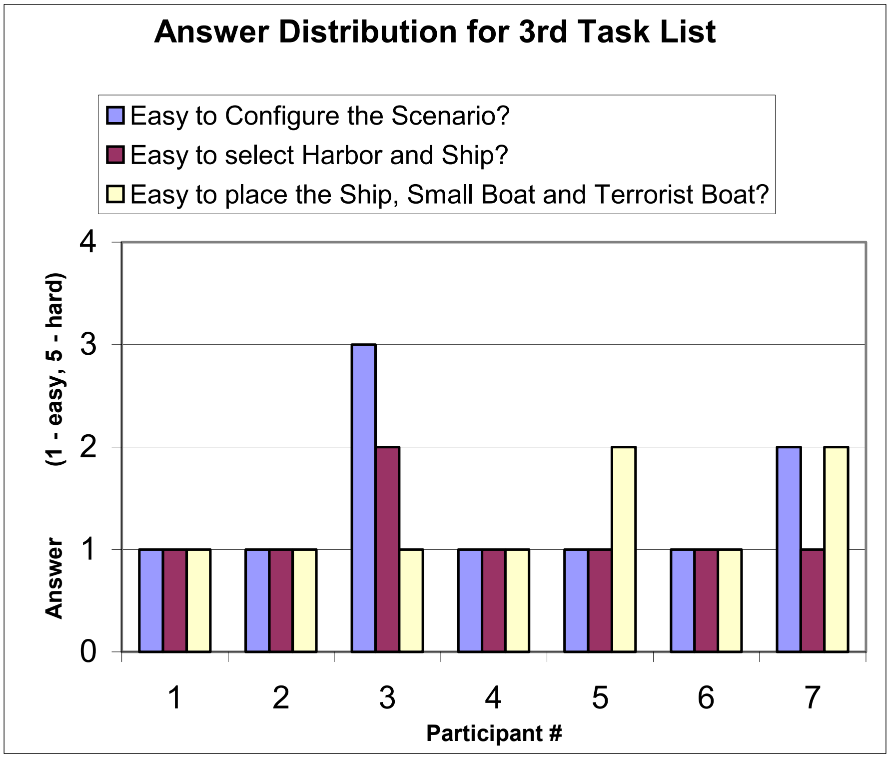


Figure 46. Answer Distribution for the 3rd task list. Level of difficulty in preferred mode (Manual or Wizard)

Notice that all answers after the third time configuring the interface are 3 or better, with the highest density near the "very easy" end of the spectrum. The middle-of-the-road "3" response was characterized as "had to think and experiment." Despite the increase in relative difficulty observed between the chart for task list 2 and that for task list 3, the majority still found the application interface obvious after the 3rd time using the interface.

The fourth question on the first three questionnaires concerned the wizard, and we consider the participant answers to the wizard questions in one combined graph (Figure 47). Participants chose their answers from the options: 1) yes, 2) maybe 3) no. The progression of the answers is interesting considering the fact that only two users used the wizard when given the choice for task list 3.

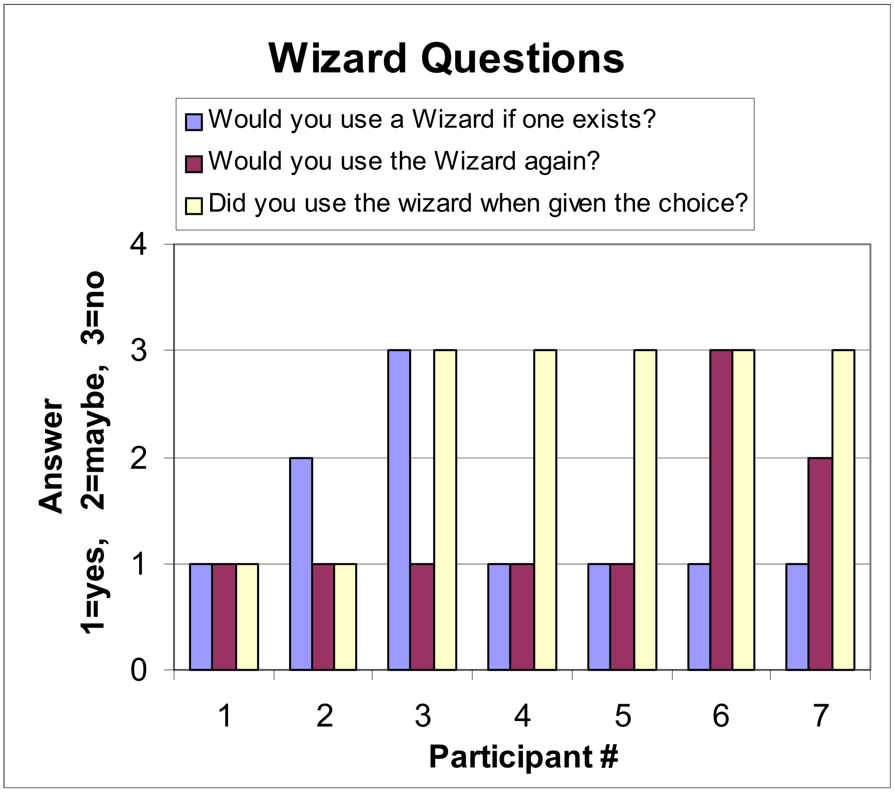


Figure 47. Depicts the wizard question distribution.

The fourth and final questionnaire asked the participants to rate their impression of some normal GUI tasks as well as manipulation of the 3D environment (Figure 48). Because of a mistake made while collecting the data, the chart is not very accurate. Half of the participants where given a draft version of the fourth task list, and all participants answered the same questionnaire, so the results are not as consistent as the previous questionnaires. This data is not as consistent for this measure as a result, but is presented nonetheless for completeness. Again the 1 to 5 scale is used with 1 being the most positive response (i.e., very easy, or vitally important).

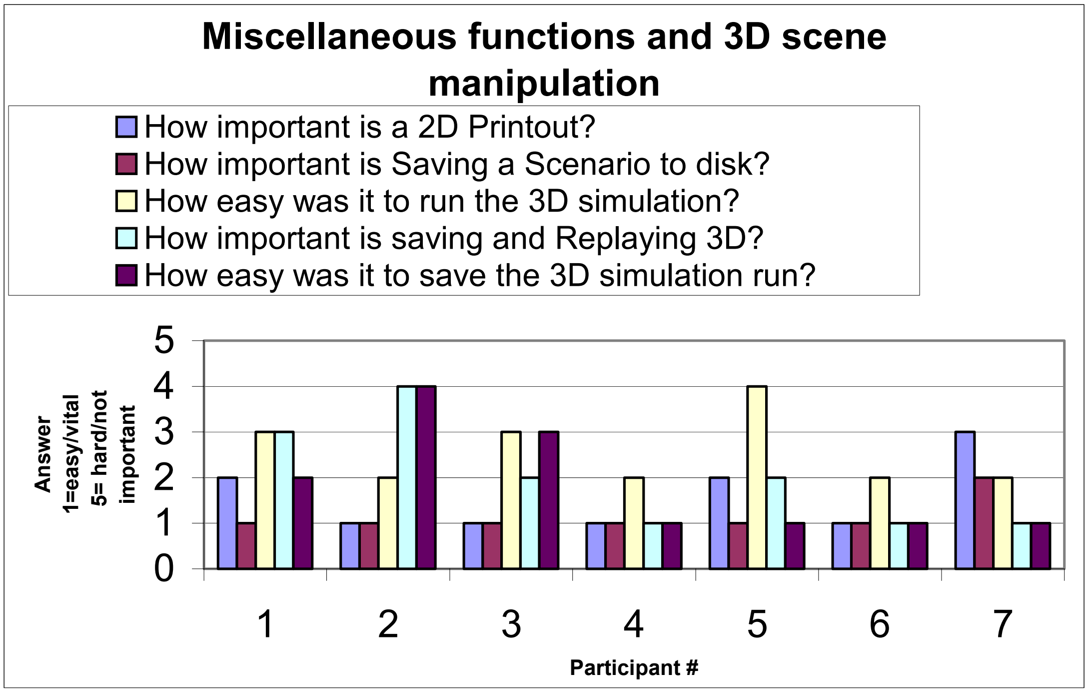


Figure 48. Depicts the chart showing the Miscellaneous functions and 3D scene manipulation.

Analysis of additional measures of effectiveness follow.

Measure of Effectiveness 2: Formatting of Results of simulation should be user-friendly and meaningful.

Only visual results were incorporated for the purpose of this test. Each end-user was able to view and interact with a 3D X3D/VRML scene in an application-invoked instance of Netscape 4.78 displaying results for which he had planned both the defenses and terrorist boat attack profiles. Users were able to compare this setup and the terrorist boat profile with their 2D plot as desired. The defense manipulation was not tested other than for scene examination and viewpoint changing. The majority of end-users found that the 2D planning tool in itself was useful; addition of a 3D capability was considered even more useful. More than one test subject mentioned the lack of any organic modeling and simulation technology for any warfare area on U.S. navy ships, and wished they had more tools besides the application we demonstrated.

Measure of Effectiveness 3: Feasibility of use as a planning tool as opposed to traditional 2D Pen and Paper methods. Should be no harder than pen and paper and require (at worst) equal time with a better result.

Once each end-user had completed one run of the application, they found it easy to do and on-par or better than traditional planning methods. Of note, the warfare area modeled in this application has traditionally lacked formal planning and precise doctrinal development at the unit level. This historical situation might have lent further weight to the some of the test-subject's favorable comments.

Measure of Effectiveness 4: Predictable Error behavior and low error rate. Should have the ability to redo previous entries into programming.

The biggest error source found was our placement of a "done" button at the bottom of the 2D planning screen for both the defensive and offensive planning modes. All tested subjects overlooked this button and were expecting something else to indicate they were done with these panels. Also, there was the capability for redoing steps (via a Backbutton), but that capability was not intuitively obvious to the end-user.

Measure of Effectiveness 5: Memorability, built in because of expected long periods of time between uses.

Incorporation of the wizard mode eliminated most of the errors found in the untrained first run manual entry test, as well as cutting the average scene creation time in half.

Measure of Effectiveness 6: When did user expectations not match system reaction?

After choosing the ship type in manual and wizard mode some users expected a visual cue on the ship description screen as to where to proceed next (such as a 'next' button) rather than going to the top level menu for the next choice as designed. Also, some test subjects expected in the manual mode that when choosing File->New. The interface would bring up the configure location option automatically similar to how Microsoft Word does with a new word document. Also, we had an inconsistency in menu naming that confused a few users. In the view submenu we listed View Harbor,

but in the configure menu had Configure->Location which did not make the task of selecting a harbor intuitively obvious to them.

Measure of Effectiveness 7: Efficiency. Do users perform the tasks in an optimal method? If not, why?

3D viewpoint manipulation was far from optimal on the first 2 viewpoints we asked the end-users to manipulate. By the third task in this area, they had learned the interface well enough to navigation the 3D scene efficiently.

Manual Scenario creation mode was not optimal. Primary reason appeared to lie in the lack of visual cues for being complete with a current step.

The manipulation of the Ship/RHIB/Terrorist Boat heading on the 2D planning screen was the primary reason for any mouse-keyboard-mouse shifts in the planning process. The majority of the users figured out that it was faster to enter the heading via keyboard directly than using our self-created 'j-spinner-like, jdk1.4' widgets. Reason for this may be that we only incremented/decremented the headings by 1. Increasing to 5 or 10 degrees, or letting the user toggle the increment, works better.

Real-time indication of changes in ship's heading appeared to be an easy area to effect improvement in the UI. The input box required a 'go' button be clicked to effect a change in any of the ship's headings. This seemed confusing to many users. They wanted a real time indication as implemented in the Detection/Non-Lethal Engagement/Engagement Range circles and the go buttons eliminated.

Other than the above-mentioned items, the only task mentioned that the users wished was easier was recording a scenario. The design did not streamline the incorporation of this at test time and this criticism was therefore expected. At test time a separate frame was brought up to record, where instead it should be embedded within the Run and Record selection.

Other than the keyboard manipulation for ship's headings the following interaction techniques were observed and planned for during the testing:

Mouse Manipulation –primary method planned for and utilized during the testing.

Keyboard Manipulation of 3D scene—All users initially tried, and most preferred, the mouse for changing viewpoints and scene examination.

The only discrepancies noted were with the switch from mouse to keyboard for heading change, which was noted above.

Measure of Effectiveness 8: User Delays.

In the wizard configuration the only delays noted were expected with the user placement of the defensive RHIB boat and planning the terrorist boat track. Not much to be done to reduce the 'tactical thinking' side of these delays other than the possible changes to the jSpinner implementation.

In manual configuration, user delays were experienced in finding the correct top-level menu selection. These could be minimized in a few ways: giving a 'next' button choice at each level in the simulation creation as well as implementing a demo mode that walks through all the steps and shows 3D scene manipulation.

Measure of Effectiveness 9: Errors.

The users did make errors during the testing. The following is a general summary:

<u>Top-Level Menu Selection</u>: First test performed with no user training conducted was in manual mode with a task list to see if the menu structure was intuitively obvious.

Some users selected File-New to create a new scenario, others did not. An engineering solution would be to implement the Mediator pattern making unauthorized top-level menu selections unavailable thereby forcing the user to select a new scenario or load a pre-existing one.

<u>Unsure of completion of a task in manual mode</u>—Some users were unsure what to do when they had completed tasks such as selecting location, ship, etc. Engineering solutions: 1) implement a next button on these pages in manual mode that takes the user to the next step; 2) make the desktop pane more like PowerPoint. When file new is selected, all required frames are iconized in order on the Desktop. The user can click on available ones to edit. Frames requiring info from previous ones might then be visually inactive and pop up a dialog stating required steps before editing. If a frame were

maximized or set to fill the entire screen, put a scroll bar on the right which, when slid, would slide from frame to frame similar to normal scroll bar behavior.

3D Viewpoint Manipulation: No user had 3D scene manipulation experience and needed time to learn the manipulation. Engineering solution: incorporate visual screen snapshots and an interactive (possibly audio-on) test scene 3D training mode explaining the manipulation before running the simulation. For the experienced user, make the first dialog brought up ask them if they wish to view these or not. Maybe also put a configuration option in that allows hints to be brought up or not in manual mode avoiding the ‘death of clippie’ syndrome.

Unsure of completion of 2D planning tasks in both modes: Most users were unsure what to do when they completed planning, some users were unsure whether they were done. Engineering solution: Steps mentioned above for the task completion in manual mode as well as giving a status check box frame on the desktop that visually indicates completion of a step with a check. This might be a little tricky, for it is difficult to determine when a user is done moving the ships, waypoints, etc around the screen. Perhaps a time-out following lost movement can pop up a hint dialogue.

All errors discovered during this test can either be engineered out or accounted for with dialog boxes. The only step that needs to be added for catastrophic errors is incorporating a “do you really want to exit” dialog when the user closes the desktop or (internal frame) manually, in order to avoid unintended loss of data.

Critical Incidents:

The primary critical incident observed was the lack of an intuitive interface for indicating to the users what to do when complete with placement of defenses or terrorist boat configuration. Some users would close the frame; some would open the next frame, etc. In re-design this was approached iteratively over several different ideas until the design of leveraging the Mediator design pattern to expose only legally available next steps in the menu selections was decided upon and implemented.

Learnability/Memorability:

The users found the system easy to use after one setup run with the application. 3D manipulation seemed to take manipulation of 2 different viewpoints before the learning factor had been achieved. A few users mentioned that given a user's manual and a fully implemented help feature, they would have no problem running the application with no prior formal training.

Efficiency and Error Rate:

The error rate dropped noticeably as a user's familiarity of the system increased. Time for completion of configuration of a scenario was noticed to increase when a user chose to use the manual method for a third test, but their statistics were still better than the first manual mode test conducted. The wizard mode significantly reduced errors compared to the manual mode. The users that chose to use the manual mode for the third run did so because they believed it would be fasted than the wizard since they 'knew' the application, but none were able to out-perform their wizard mode test statistics. Interestingly, even with this knowledge afterwards, they still preferred the manual mode.

Satisfaction and Conclusions:

The majority of users were satisfied with the system and would like to see it online for real usage. We believe this is indicative of the lack of deployment of modeling and simulation technology to the tactical ship level in the fleet today despite the pre-existence of both the hardware and software to support applications such as the one we have developed. Although these tests performed minimal testing on the 3D interface since the design testing focus was on implementing the 2D planning interface to the model, contradictory results were found to those in a previous study of the VRML based Capture the Flag game-- users actually enjoyed the 3D manipulation in our application and did not want their control of the scene limited although there is a demonstrated steep learning factor with the 3D plug in interface in internet browsers. This appears due to the fact of having a very focused user-base from the conception of the project, all of whom have (and see) a need for the application developed, vice just seeing it as a game where they may or may not be attuned to specifics of warfare specialties not their own (i.e. the

difficulties experienced by users during the CTF tests trying to drive a tank or helicopter when they were unfamiliar with the control names, etc.).

Probably the best design input beyond the errors noted during the usability testing were the comments provided by our test subjects for possible corrective engineering solutions regarding problems they had running the simulation. As a result, a minimum of one and probably two more rounds of usability testing need to be conducted, if development of the AT/FP application continues. Specifically, future testing needs to entail similar experiments as ran here, but must also focus more on the 3D interface and analytic results from the incorporation of intelligent agent technology.

Finally, the only item that might hold back real-time deployment to the fleet within nine months the identification, development, and integration of 3D harbors for the application's widespread use. Auto generation of basic 3D is possible from the available 2D chartlets, but extensibility for other threats than small boats would require the development of surrounding land and structures in the harbor areas through contractual support or other project expansion. As a result, the final application is implemented for Port Hueneme California, Aden Harbor Yemen, and Pearl Harbor Hawaii.

K. USER INTERFACE REDESIGN

It was determined that no major redesign of the conceptual design was required, only specific improvements in the interactions and interaction methods of specific portions of the application identified by the testing conducted.

Main Panel:

--In manual configuration mode, when menu choice File->New is selected, the configure location dialog is displayed to the user.

--Tactical data Filename is placed at the top of the main frame, as well as for the applicable internal frames for the scenario being configured.

Configure Location Frame:

--Include a next button at the bottom of the frame for the user to select when ready for the next configuration step.

--Once the next button is selected, the frame will be iconized and placed in upper left quadrant of the application desktop.

Configure Ship Defenses Frame:

--Ship and RHIB headings are updated in real time as the user manipulates. Go button that was previously required to actuate changes is eliminated.

--Set Ranges button eliminated since the ID, Non-Lethal, and Lethal Engagement ranges a already updated in real time by user input.

--JSpinner buttons for ship and RHIB heading manipulation now increase/decrease the heading by 5 degrees vice 1 degree to minimize the number of mouse to keyboard to mouse transitions.

--Next and Back buttons replace the Done and Cancel buttons at the bottom of the screen. They will be in the shape of arrows. When the user is complete with the configuration he will select the next button and the defense configuration frame will be iconized in the lower left quadrant of the desktop.

Configure Terrorist Boat Frame:

--Terrorist Boat Heading updated in real-time.

--Spinner's have the same increment/decrement range as for the ship/RHIB heading manipulation.

--Same changes for Next/Back buttons and the elimination of the go button.

--Waypoint numbering changed to count starting with one vice zero waypoints, and the number of waypoints will be updated in real time as they are selected on the GUI.

--When next is selected the current frame will be iconified in the lower right quadrant of the desktop.

Ship Selection Frame

--Next and back buttons incorporated as mentioned above

--When iconified will be in the upper right quadrant of the desktop

--Shall contain a link to additional unclassified ship class information that will be 'googled' locally from either Jane's online or the Federation of American Scientists web page (www.fas.org) (accessed February 2003)

All four main frames will then be iconized on the desktop. The user will be able to select one to reconfigure if desired. Due to the reliance of follow-on frames on their predecessors for simulation configuration, the user will then be forced to at least acknowledge via the next button the change in configuration on the resulting frames.

If multiple new files are opened (not tested previously), then the last one opened will be on the desktop and the user will need to select the previously opened one via the file menu to switch.

L. UI RESULTS AT COMPLETION OF THE FIRST ROUND OF USABILITY TESTING

The UI look and feel improvements resulting from this first iteration of the usability process for the ATFP Scenario generator resulted in the look and feel depicted in Figures 49-54.

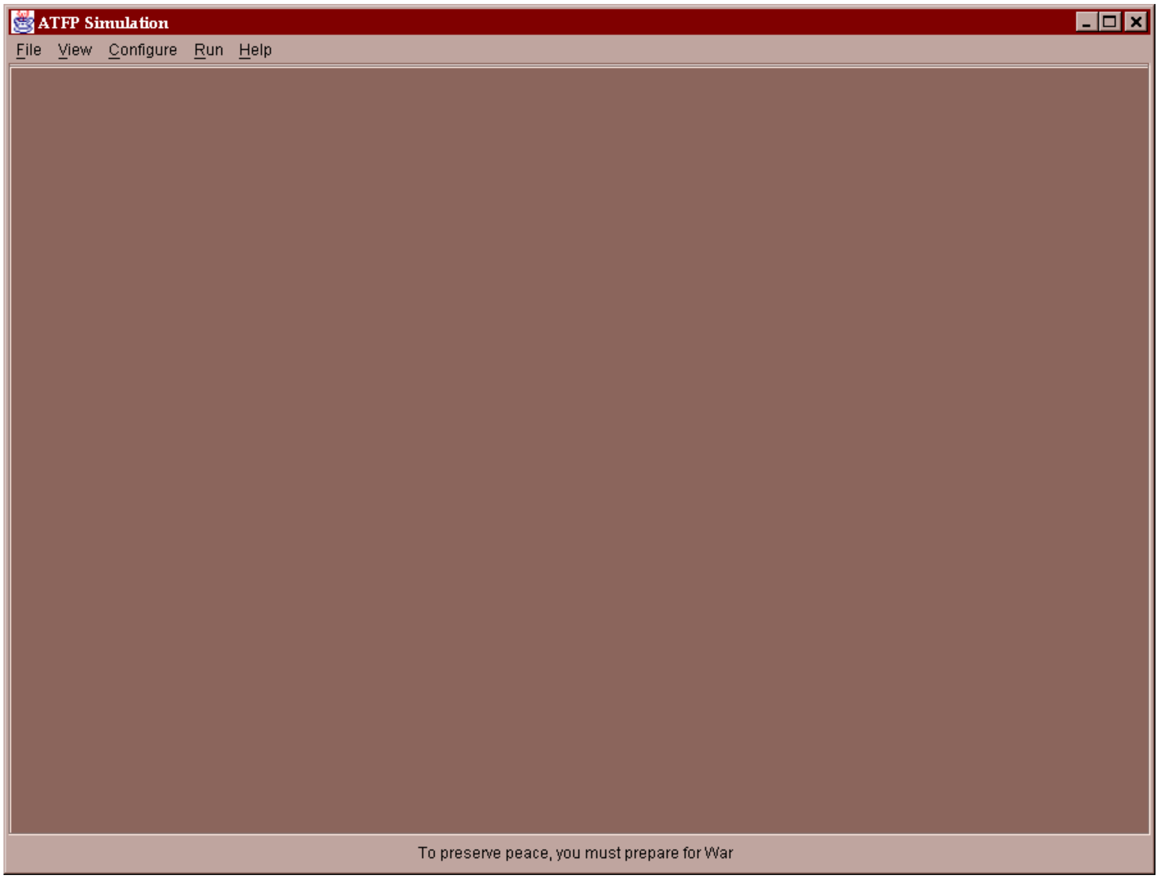


Figure 49. AT/FP Scenario Generator Main UI Screen from March 2002.

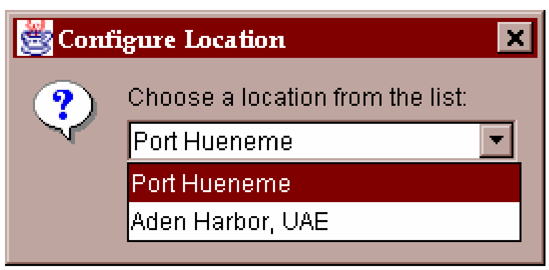


Figure 50. Depicts Harbor Location selection screen.

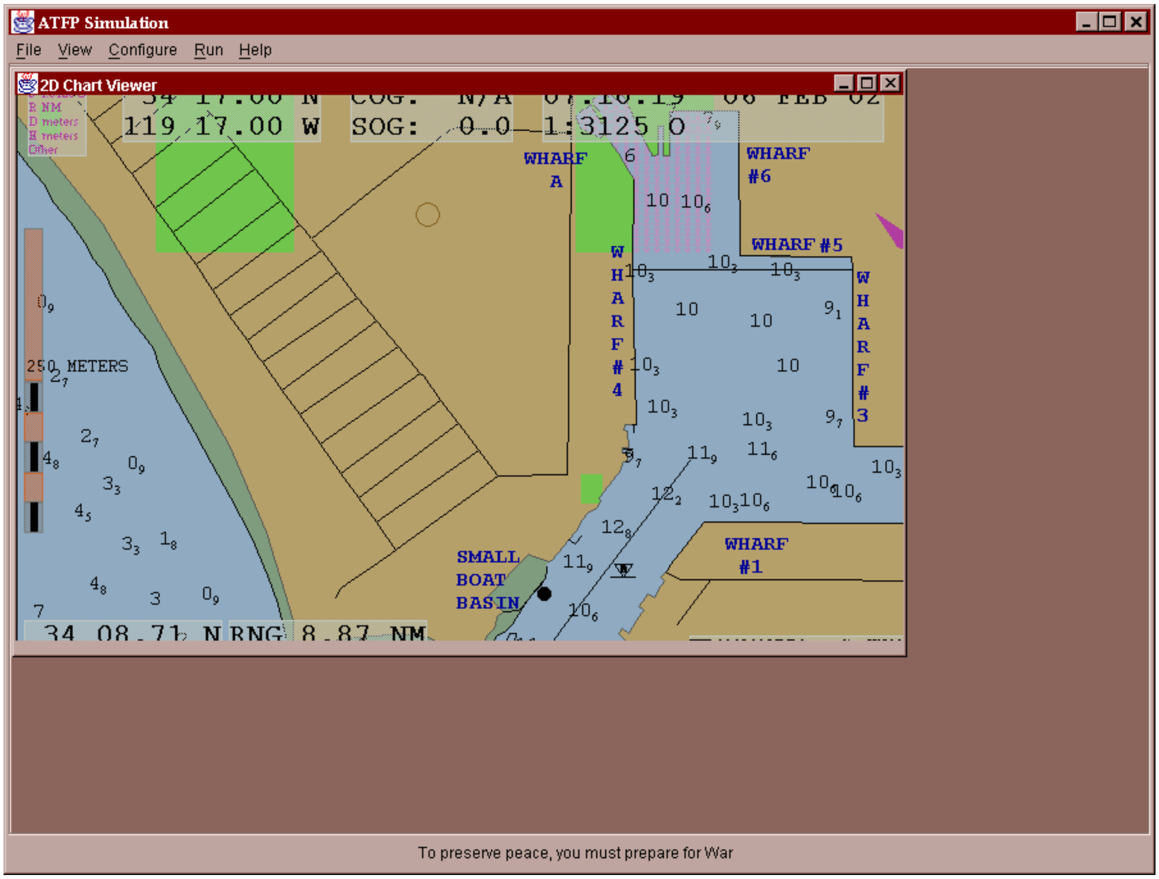


Figure 51. Depicts 2D Chart view of the harbor selected.

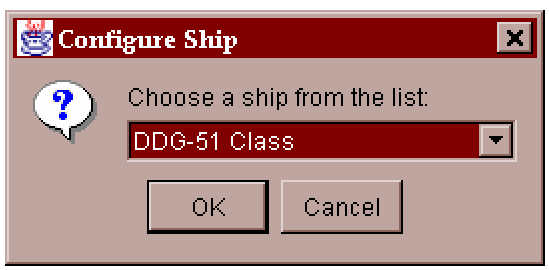


Figure 52. Depicting the ship selection screen.

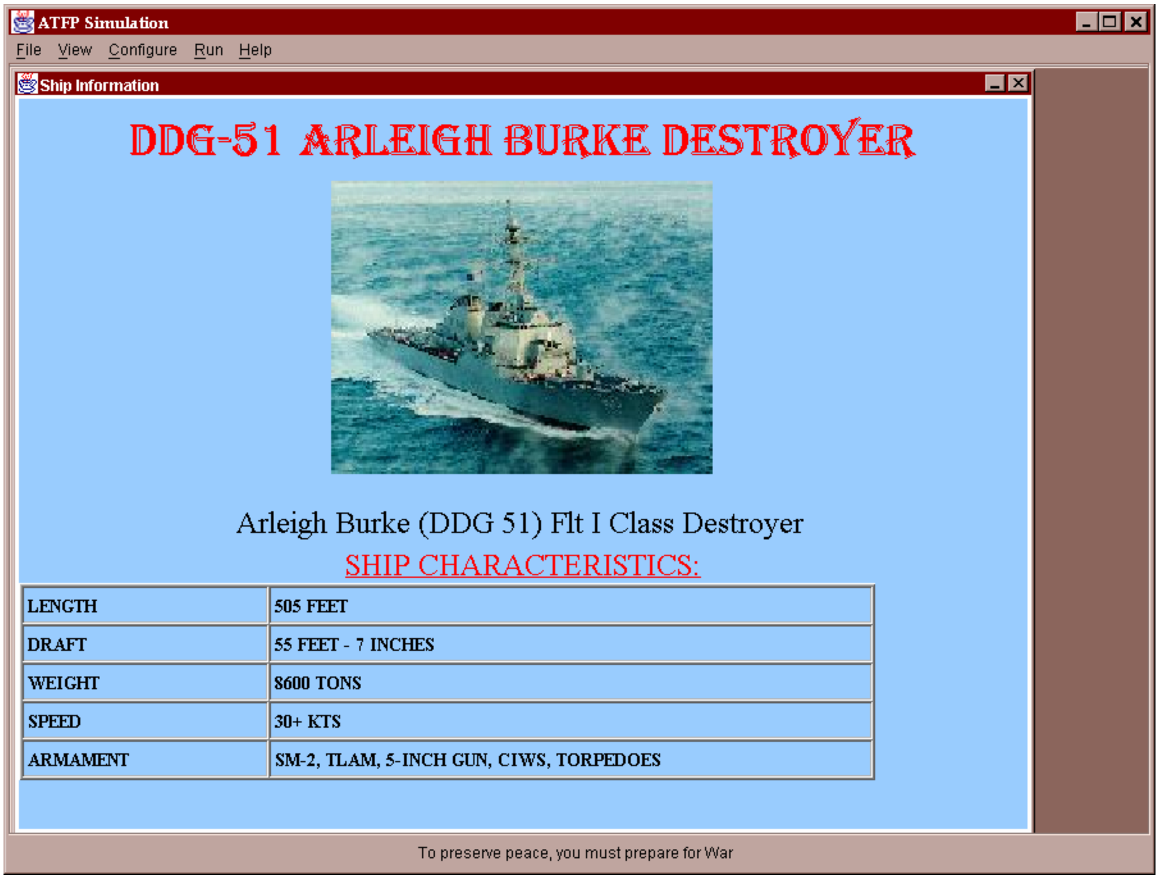


Figure 53. Depicts the Ship Information screen, after ship selection has been made by the end-user.

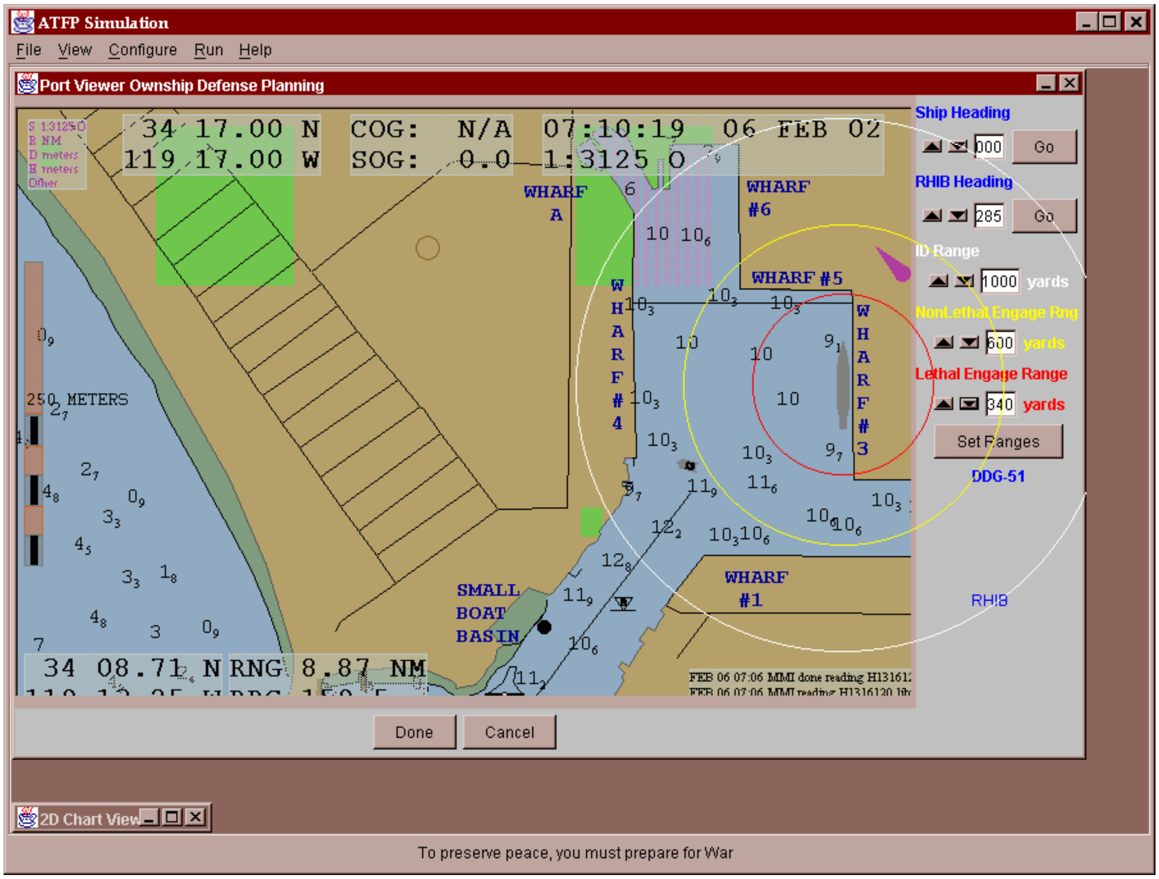


Figure 54. Depicts the Defense configuration screen.

M. SUMMARY

By leveraging industry standards towards usability, a more memorable, usable, and overall higher level of quality application interface was designed and implemented in less time than would have been possible if not using these processes. Although time has not permitted additional iterations of this process on this application before completion, it is recommended that anyone following this work do so. Additionally the reader is referred to the recently created IRB process at the Naval Postgraduate School for current guidelines and procedures necessary for effecting additional studies.

THIS PAGE INTENTIONALLY LEFT BLANK

VII. ANTI-TERRORIST/FORCE PROTECTION MODEL DESIGN

A. INTRODUCTION

This chapter outlines the general object model design performed in the execution of this study. Specific attention was paid to the leveraging of XML technologies to maintain a strict separation between models and the various graphics views.

B. SCENARIO OBJECT MODEL DESIGN

Much thought was spent in designing an extensible scenario object model to facilitate future additions and modifications of required data needed to run an anti-terrorist scenario. The primary ideal was that an abstract base ScenarioComponent class can be developed that is be extended by specific entities within the simulation system. These extended entities might then be contained as composite objects inside of a primary ScenarioClass object which also contains ScenarioStatistics and ScenarioRun classes responsible for the gathering of statistical data for a given scenario run or run(s) initiated by the end user. These components are listed in Figure 55.

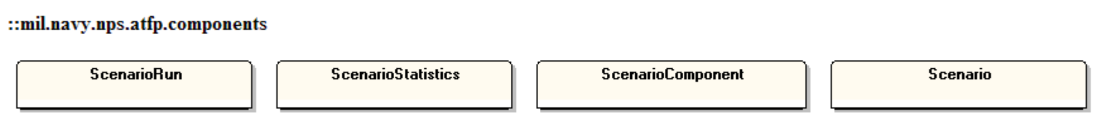


Figure 55. UML Diagram depicting the atfp.components java package members.

So, building upon this abstract set of relationships, entities within the simulation system can correspond to either surface entities or the land, where the land object contains all other non-organic, or static objects within a given scenario. Entities include a terrorist or terrorist(s), neutral(s), defensive boats, and high-value unit(s) with special attention paid to the extension of the application towards other warfare areas at a later date. Refining the Scenario Component ideal further, the various scenario components are represented by Entity types shown in Figure 56.

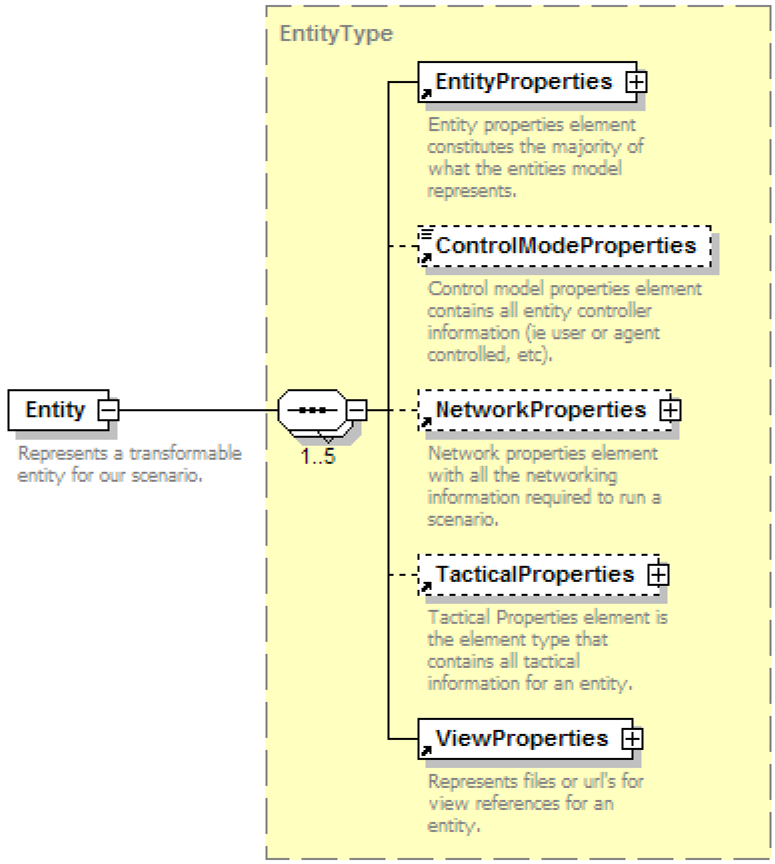


Figure 56. The Entity Data Type that extends the idea of Scenario Components for representing entities in our scenarios.

As you can see, the entity type has five categories of properties that can be associated with it split roughly according to the Model-View-Controller design paradigm. Control properties are separated into three sub-areas: ControlModeProperties, NetworkProperties, and Tactical Properties.

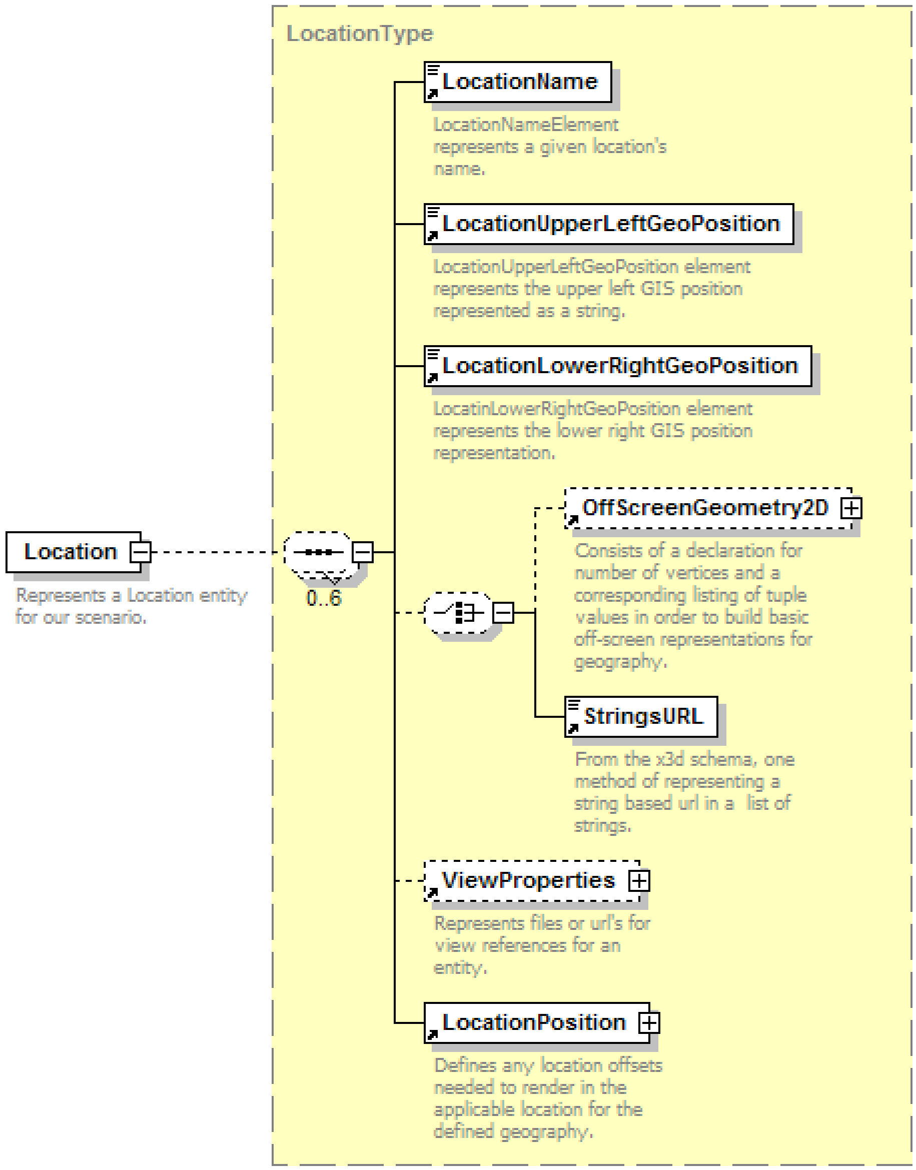


Figure 57. The Location data type that extends the ScenarioComponent class in order to represent a harbor and other inanimate objects in our simulation system.

The Location type (Figure 57) is split up in a similar manner, except it contains additional markup information that will likely be required if trying to interface directly to a Global Information System (GIS) based C4I or simulation system at a future date for future work including reconfiguration or stylesheet conversion to alternative formats such as the North Atlantic Treaty Organization (NATO) Generic Hub XML database format.

Of particular interest is the flow of control for the scenario once started. Once the end-user has configured all of the required scenario components, an EntityController

class is used to manage our scenario runs. This occurs regardless of whether they are agent driven, user driven, or a combination of the two. The EntityController object starts all of the specific entity threads, and manages the process of both inter-thread communication as well as managing the opening and closing of network sockets to send DIS packets that update the entity views, as well as to receive any information for entities that might be provided in the scenario from a networked player. The architecture for the EntityController is based on CMAS architecture laid out by [Osborne 2002], where the delivery of entity state information is one of the primary connector based inputs for sensory information delivery to entity actors/agents in our simulation system (discussed in greater detail in the next chapter). By utilization of this socket-based communication infrastructure for manipulating the view of our scenario, the view is enabled to be anything we would like to define, and through the utilization of a well-defined and known network protocol in DIS, the same simulation architecture can be reused to render a 3D view in a pure X3D system such as a web-browser plugin, Xj3D Browser, a modified one such as NPSNET V, or even an unrelated to X3D viewing system such as a proprietary based graphics engine. For this thesis, NPSNET-V and Xj3D were utilized as the primary 3D graphics views, but by keeping an agnostic view on the graphics requirements make the project less dependent on any one graphics standard or implementation.

C. REPRESENTING SCENARIOS IN XML

Now, since the entities in the scenario object model have been defined in an extensible and straight forward manner, it makes sense that the capability to save and read scenario instances from disk is desirable. XML documents were chosen to be the basis for this for a few reasons. First, the ability to leverage XML stylesheet technologies to render varying 2D and 3D graphics views of the scenario instance files was a huge advantage. Secondly, since the instance documents are based off a well-defined schema, it is possible to edit scenario files with a text editor or another XML document editor and produce valid content outside of this application. The JDOM open-source API for XML processing in Java was chosen as the primary XML API to utilize for serializing to and from disk in this application. The SAXON 2.0 XML API was utilized with JDOM to read files from disk, and the DOM XML API was leveraged to apply XML stylesheets

within the application context. The IBM Dom4J API and Sun's JaxB data binding API were examined but not used in this version of the application work. The syntactical requirements for applying any of these programming libraries proved to be straightforward to apply by utilizing the in-depth examples provided by the library developers.

D. VISUAL DISPLAY VS OFF-SCREEN MODEL REPRESENTATIONS

A necessary feature to truly separate the scenario model and controls from the graphical views was creating an adequate model representation for entities and harbors when operating off-screen vice their visual display. This basically consisted of items such as an entity's dimensions, physics parameters, and other control parameters. For off-screen representation of land, the problem proved to be a bit different than for terrain following and collision detection for land-based entities in that waterborne entities had to be able to tell when collision with any land or another entity was imminent and provide basic reactions to these occurrences. As a result, the java.awt.geom.Polygon class was utilized to provide first order basic bounding box checks for these types of collisions with good success (Figure 59).

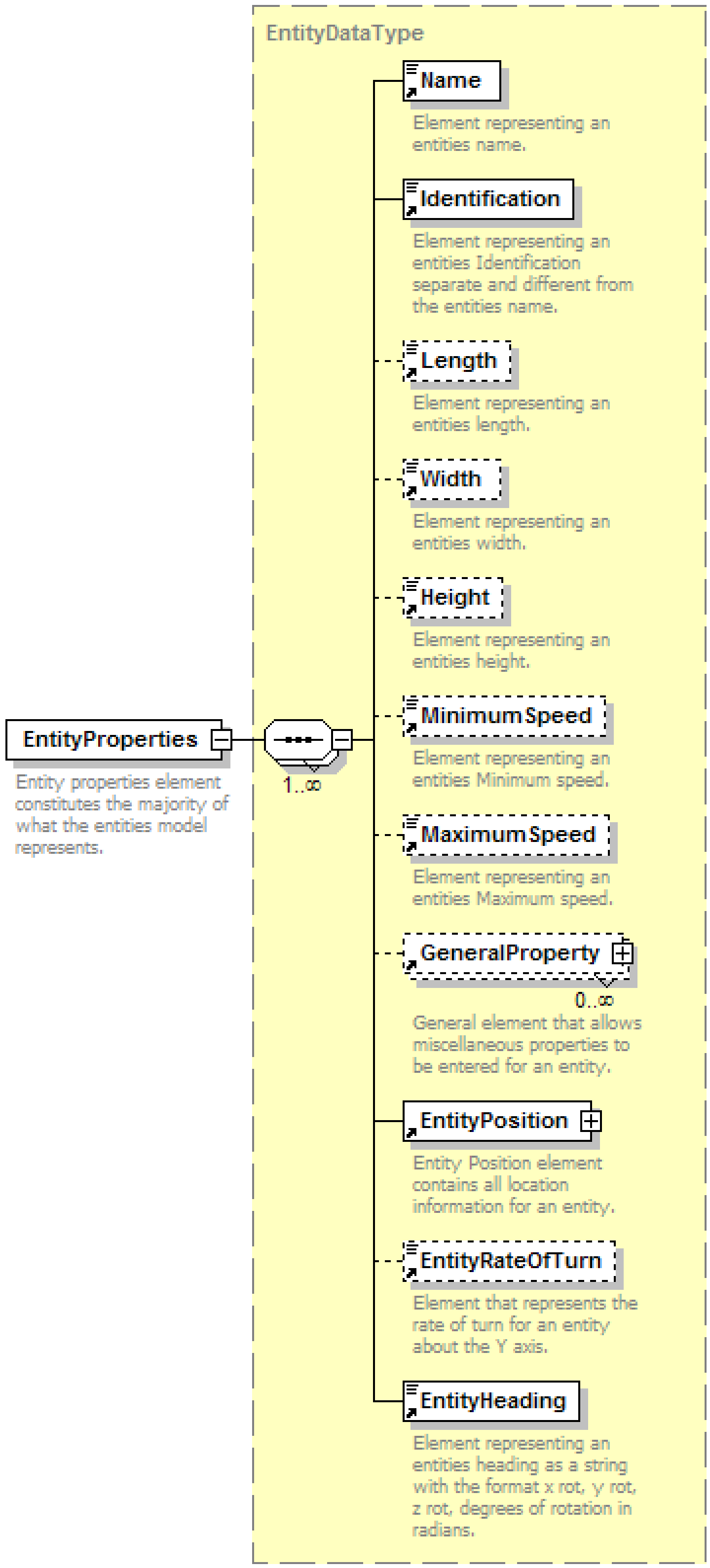


Figure 58. XML Schema design view of an Entity's properties for the AT/FP Scenario Generator application.

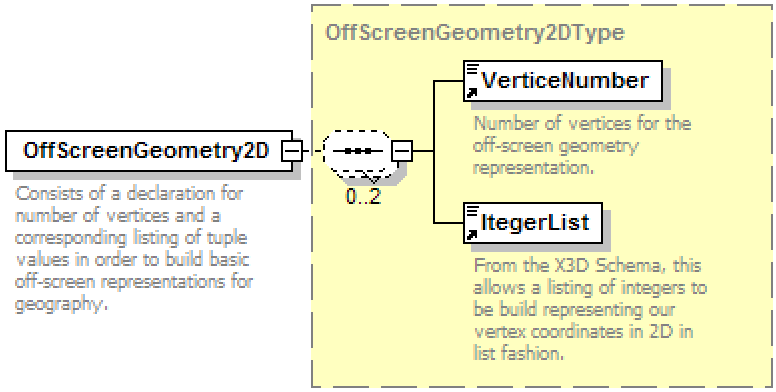


Figure 59. XML Schema design view of how the bounding box coordinates are defined for storage within a scenario XML instance documents.

E. DYNAMIC SCENARIO GENERATION UTILIZING XML AND XSLT

As shown in [NICKLAUS 2001], XML and XSLT have been used before to dynamically create X3D graphics scenes from XML based Operations Orders. In this case, however, the application wants to dynamically create 3D scenes, 2D statistical graphs, and XHTML 2D post scenario run feedback. The basic construction process for all of the different types of views that might be created via stylesheets consists of:

1 - Walk the XML instance document tree.

2- For each node, apply a modular XSLT template to output specific XML matching a predefined component such as the syntax for displaying a location or ship in 3D, or creating an SVG Polyline plot, or 'pretty-printing' the scenario information in XHTML.

For example, consider a high level view of an instance document such as that given in Figure 60:

```
<?xml version="1.0" encoding="UTF-8" ?>
- <ATFPScenarioType>
    - <Head>
        - <TacticalData>
        + <Location>
        + <Entity>
        + <Entity>
        + <Entity>
        + <Entity>
        + <Entity>
          </TacticalData>
      </Head>
  </ATFPScenarioType>
```

Figure 60. Example high level ATFP Scenario XML Instance Document

We can look at a few XSLT snippets and the resulting views. First, to create a 3D view, the XSLT starts with:

```
<xsl:template match="/">
    <xsl:call-template name="npsnetBuilder"></xsl:call-template>
</xsl:template>
```

Figure 61. Example XSLT template invocation.

which just says to start with the root "/" of the document we've loaded, then call the other defined templates contained within our stylesheet. So, one example of a subset of one of the templates called within this stylesheet is listed here:

```
<xsl:for-each select="ATFPScenarioType/Head/TacticalData/Entity">
      <xsl:choose>
         <xsl:when test="EntityProperties/Name/. = 'DDG-51' ">
         <ccei:Entityname="org/npsnet/v/entities/platforms/surface/DDGArleighBurke.xml"
            modelName="high_value_unit">
            <ccei:Transform> <xsl:attribute name="rotation">0 -1 0
            <xsl:value-of select="EntityProperties/EntityHeading"></xsl:value-of></xsl:attribute>
            <xsl:attribute name="translation">
            <xsl:value-of select="EntityProperties/EntityPosition/XYZPosition">
            </xsl:value-of></xsl:attribute> </ccei:Transform>
         </ccei:Entity>
         </xsl:when>
         <xsl:when test="EntityProperties/Name/. = 'DD-963' ">
         <ccei:Entity name="org/npsnet/v/entities/platforms/surface/DestroyerSpruance.xml"
            modelName="high_value_unit">
            <ccei:Transform> <xsl:attribute name="rotation">0 1 0
            <xsl:value-of select="EntityProperties/EntityHeading"></xsl:value-of></xsl:attribute>
          <xsl:attribute name="translation"> <xsl:value-of select="EntityProperties/EntityPosition/XYZPosition">
          </xsl:value-of></xsl:attribute> </ccei:Transform>
         </ccei:Entity>
         </xsl:when>
```

Figure 62. Example XSLT template demonstrating the use of xsl:for-each and xsl:choose for creating dynamic 3D scenarios.

The resulting 3D scene is shown in Figure 63.

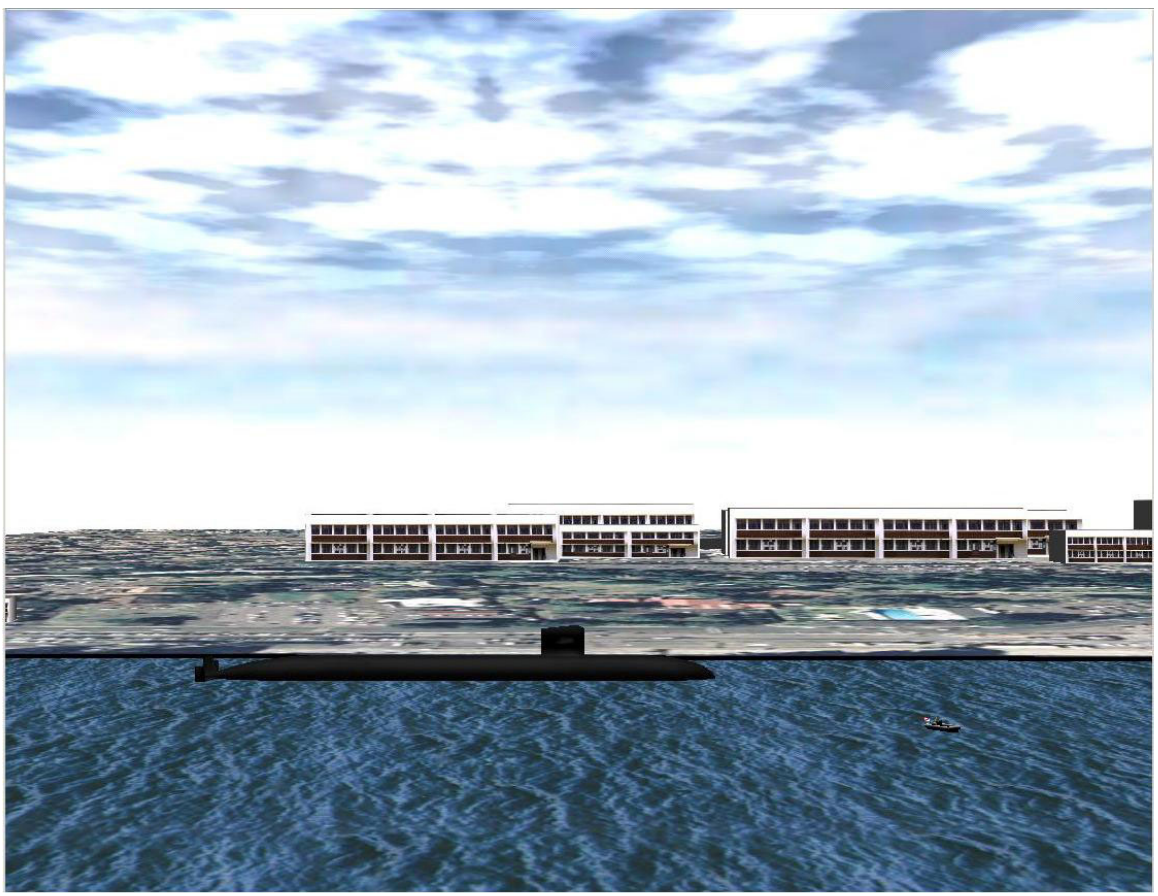


Figure 63. Example 3D Scenario dynamically created through the application of an XML stylesheet against a scenario instance document.

Similarly, in Figure 64 we see a rendered example of the application of XML stylesheets for SVG displaying a multiple-run for statistics:

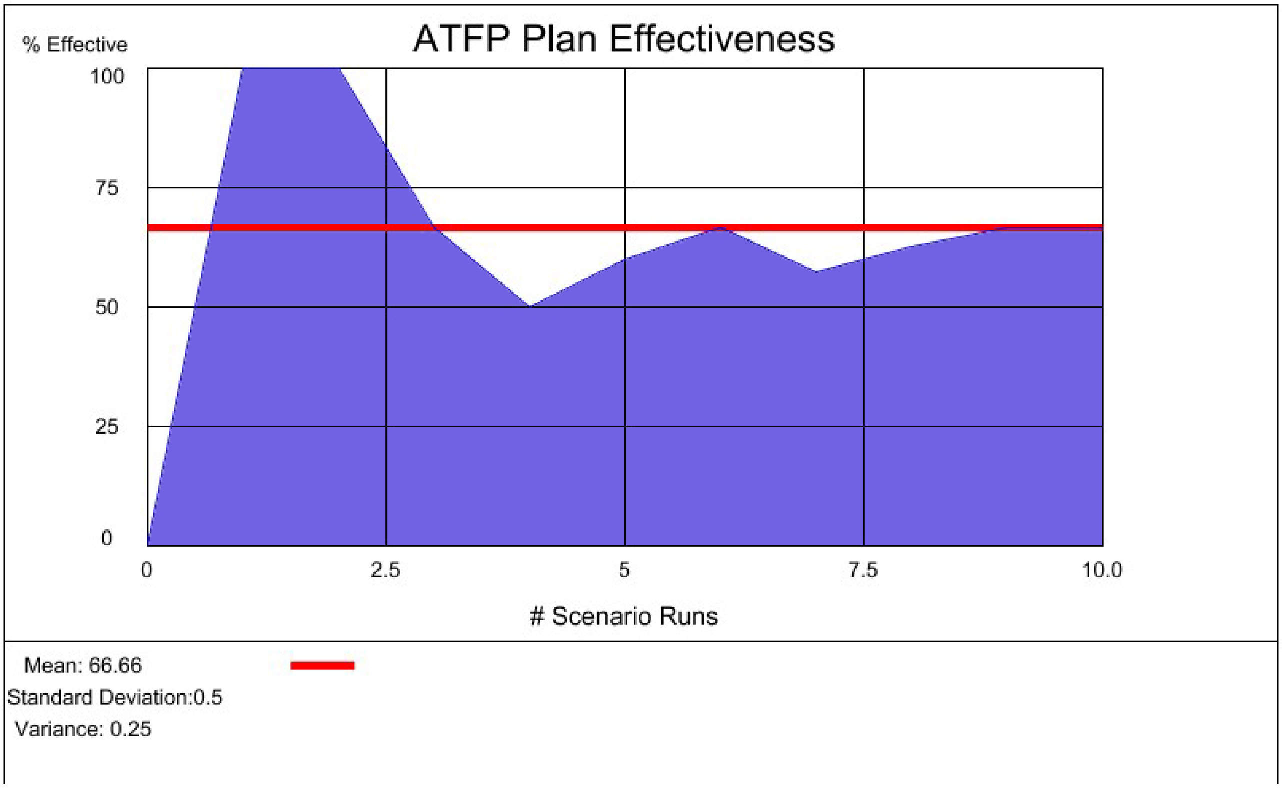


Figure 64. SVG depiction of a scenario run for statistics.

Finally, in Figure 65, we can see an example of XHTML that is stylesheeted from a single scenario run for 1 run feedback to the end-user.

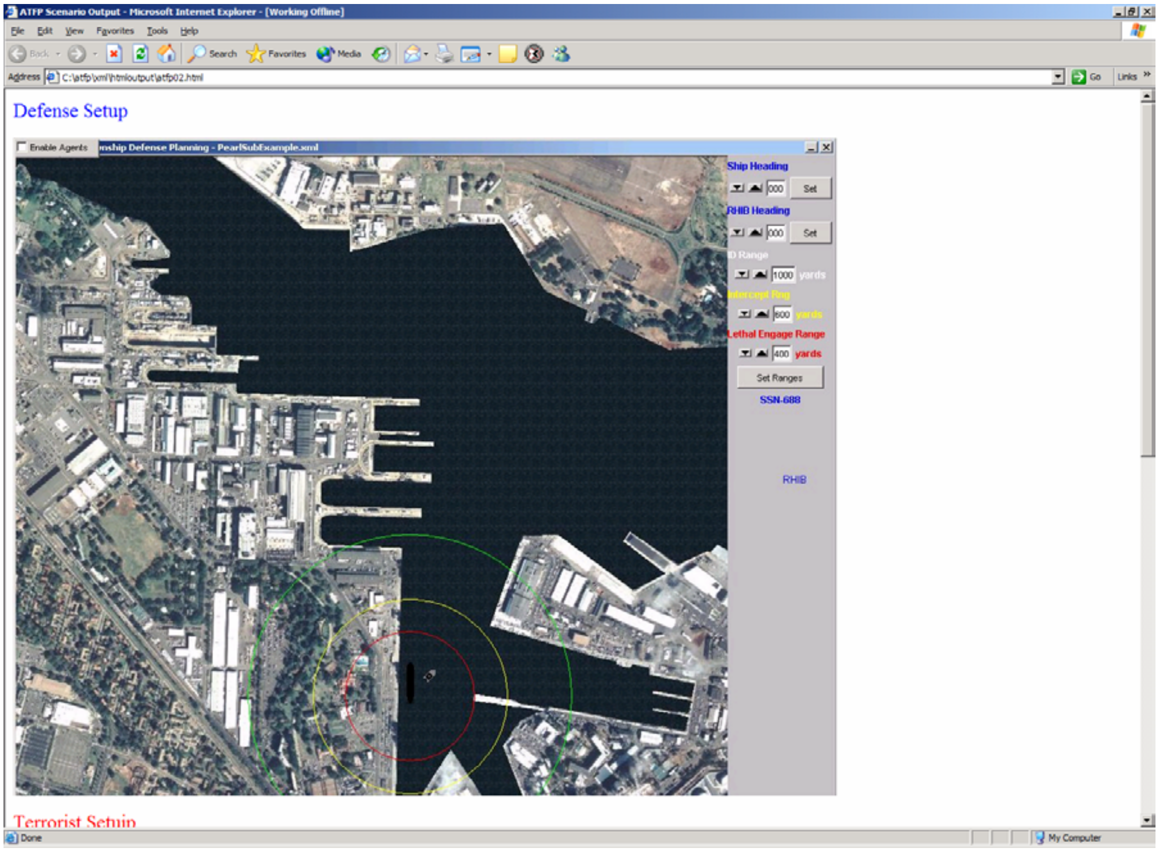


Figure 65. XHTML Scenario Feedback slide stylesheeted from an XML instance document.

F. SUMMARY

This chapter exposes the basic design methodology utilized for representation of the AT/FP Scenario object model, and the various ways XML technologies are leveraged to render varying views of the resultant XML documents from an end-user configuration.

THIS PAGE INTENTIONALLY LEFT BLANK

VIII. AGENT DESIGN AND IMPLEMENTATION

A. INTRODUCTION

This section provides an explanation for the methodology carried out whilst designing and implementing a multi-agent system (MAS) for this thesis. Agents implement situated logic directly corresponding to friendly and hostile tactics in order to allow the targeted end-user to gain greater tactical insights., and also to enable exhaustive tactical evaluation to automatically assess tactical Measures of Effectiveness (MOES) while highlighting particularly hazardous encounters.

B. USING A MULTI-AGENT SYSTEM TO GAIN INSIGTS ONTACTICAL LEVEL OF WAR

Typically employment of doctrine at the tactical level of war for navy ships occurs more in the discussion/experienced based context of individual wardroom or battle-group level staffs. Unfortunately it generally involves little to no on-site analysis of programmatic what-if's against the warfare or ship commander's plans. The high level goal for the utilization of autonomous agents in the context of this research is to apply agent technologies to show how one might be able to apply these (and traditional operations analysis techniques) to give the warfighter a better 'toolkit' with which to employ doctrine while going in harms way.

Notably this slows quantitative assessment of the specific problems posed for utilizing agent technologies to gain insights include:

1) Analyze Chief of Naval Operations (CNO) and Court of Inquiry (COI) recommendations regarding picket boats to defend against the USS COLE – Al Qaida scenario.

2) To gain insight into the advantages and disadvantages of the placement of defensive picket boats in different harbor environments.

3) To gain insight on the relationship of the distances for tactical parameters in relation to the placement of defensive picket boats.

4) To determine the impact of frequency of background shipping on the effectiveness of the tactical parameters and picket boat placement.

5) To determine possible needs for non-organic defensive support for navy ships while in-port at a high threat level.

6) To identify further areas for study utilizing autonomous agents.

C. ANALYSIS FOR APPLYING AGENT TECHNIQUES

[Hiles 2002] details the specific design process and questions asked that were followed during the agent design for this research. First, as outlined in chapter I and by both [Ferber 1999] and [OSBORNE 02], we start the agent design process is started by applying our formula for a multi-agent system, MAS={E,O,A,R,Ops, Law}, and define each in context of our targeted environment.

1. Environment

First, the simulation environment is defined. This aspect is critical, for if we include too little information or detail, then we risk providing false insight, and if we provide too much we may not be able to gain anything. In this light, it must be observed that insight gained from any form of analysis must be taken in light of a function of the environment modeled, questions being asked, and possibilities of further parameters to repeat the scientific process.

For this context, the outer environment, or E_{outer} is defined to be the area in or about specific harbors of interest that navy ships will be berthed. The environment will consist of the geometry of the harbor with importance placed on the defining boundaries between the ocean and land or human produced structures. Building structures will be incorporated at a high level, since fine resolution of detail for these structures is not considered essential for execution of the MAS being developed. Movable objects and/or actors will be modeled as well with basic rigid-body kinematics physics models but dynamics models not incorporated to search for possible answers to the insights we are looking to gain. The E_{outer} is then thought of as being represented specifically by two

things: 1) The representation of terrain object(s), and 2) the dynamic representation of all other entities in the simulation system.

2. Objects

There are two basic object types in our environment: 1) Entities, and 2) Terrain. Terrain objects are thought of as being composite types of objects. They can be comprised of terrain, piers, buildings, nautical buoys, and so on. They can function as either obstacles or resources for entities depending on the entity's goals. Resources can be used to 'hide' movement from another entity; obstacles cannot be moved through and energy must be utilized in order to decide how to best move around. Entities are ship entities that are in the simulation.

For the agent model, the visual display quality is not important for the outcome of a scenario run or runs, but might be considered necessary to increase the immersive experience of the end-user when utilizing our system. As a result the essential attributes of terrain objects are their locale in the virtual world, and dimensions based about this locale (specifically their length, width, and height). Essential for viewing a running scenario are the attributes of the primary, secondary, and tertiary graphics representation of these objects and the subcomponent requirements of these files, in our case X3D graphics and VRML97 files. Essential attributes for entity objects in addition to those for terrain objects are the various physics components required for implementation of rigid body kinematics such as maximum speed, minimum speed, turning rate, rate of acceleration, and so on.

3. Agents

The primary actors that possess intent and can act autonomously are defending, attacking, and neutral ships. They can act autonomously, or in a non-deterministic fashion by allowing zero, one, or more to be driven by human-in-the-loop controllers compared to always maintaining total autonomy.

The elementary operations of these actors are:

1) Ability to move a given distance during a given time step in the simulation

2) Ability to increase or decrease speed

3) Ability to effect a change in the direction of movement vector

4) Basic reactions for collision with terrain objects or other entities

5) Ability of defending agents to intercept other agents

6) Ability of attacking agents to effect a TNT equivalent detonation of their entity against another

7) Ability to drive to a goal based position: for neutral entities to a given locale; for attacking entities towards the target of choice; for defending entities to intercept other entities that might be threats.

Defending and attacking entities are distributed at the start of a simulation run by end-user setup in a 2D user-interface environment. The frequency, but not location of neutral entities is chosen by the end-user prior to running a simulation. Once the simulation is initiated, entities change over time by proceeding or changing position based on their desired goals. The change in position is effected in a real-time simulation model driven by dynamic based timesteps, generally occurring about 30 time per second.

Specific decisions that situated (agent or human) entities must make are:

1) Have I collided with anything (ie a terrain object, another entity?)

2) Do I want to move? If so in what direction and at what speed is best to move in?

3) If I'm defending another entity, when do I want to move to intercept another entity? When do I cease intercepting an entity?

4) If I'm attacking another entity what is the best way to move there? When do I attack the entity? Can I be deterred? If so in what manner do I act? If I can be intercepted, what parameters are necessary for this to occur?

While effecting these decisions, the defending and attacking entities can be thought of as being in competition, with the neutral entities also competing on the basis of taking up sea space and wanting to move to a desired location in the world.

The goals of defending entities are defined as a function of the input of tactical parameters by the user configuring the simulation. At the given Identification Range, they must try to identify all surface craft entering this range. At the Intercept Range they must intercept all entities proceeding towards the defending entity. At the Lethal Intercept range they must utilize any weaponry onboard.

The goals of the attacking agents are to proceed in light of their personality towards an effective attack against the entity being defended. They can either have the attribute of being able to be deterred, attacking at all cost, or a hybrid consisting of values in-between.

The goals of the neutral agents are to proceed to their desired point of travel. If queried for intercept to either stop or continue movement until forced to stop.

4. Relationships

Relationships are defined during scenario configuration. Defending entity and applicable tactical parameters are defined. The duration is for the length of a given scenario ending either by a successful defense or attack. They are cooperative in nature. Attacking entities are also defined in the same manner but are considered to be competitive in nature. They can be cooperative with a neutral entity for purposes of concealment from the defending entities.

5. Operations

All entities can perform the operation of movement within the scenario environment. Additionally attacking entities can affect an attack through self-detonation and defending entities can conduct movement for the purposes of patrol, search, or interception of other entities.

6. Laws

There are two types of global information defined in our system. The first is the geometric layout of the terrain object(s) as represented in three space. The next is the locale, direction, and speed vectors of the actors in our system. The actors do not have omnipotent knowledge of E_{outer}, so this view will be limited based on the

entity's determined line of sight before being passed to E_{Inner}. In the same manner, knowledge of terrain objects will also be limited.

Actions based on connectors being bound at runtime may be changed by the individual agent if past performance for a given ticket or sequence of tickets is deemed to be unsatisfactory.

Entities must effect actions based on the collision with terrain objects or other entity objects to keep from violating the system parameters.

D. REPRESENTATION

The audience for this MAS are the same as outlined previously in the usability study. Specific data generated by agents include the real time based position data of all entities within the system in addition to the decisions being made. Positioning information will be depicted in real time in X3D graphics, non-real time with statistics generated from non-rendering simulation runs, as well as being able to view scenarios as they iterate over multiple generations. When running offscreen, data for scenario results are collected and displayed to the user at the completion of execution. During scenario runs, the agent brainlid's depicting what the agent thought process is can be viewed. At the completion of a scenario run, the user will be informed as to the single scenario run results, and allowed to review screen captures of the scenario configuration of defenses and attacking craft to gain insight into relationships between the various parameters.

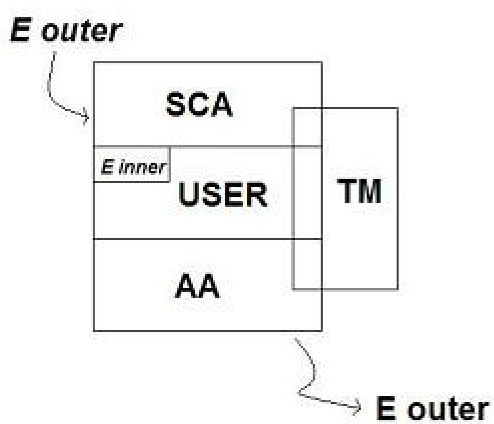
E. APPROACH

The high-level view of an actor in our system can be thought of as being rigidly separated into its model, view, and controls. Specifically, we are concerned with the control portion for our MAS system. All agents receive inputs from E_{outer} which are provided to the Symbolic Constructor Agent (SCA). The SCA then filters the input stream of information before providing it to E_{Inner}. The Reactive Agent (RA) then makes decisions on what actions it wants to effect based on the currently received feedback from the environment and any other inputs it may have on the inner environment. It then passes the stream of tickets or frames consisting of tickets to the Action Agent who de-conflicts the actions and effects the applicable ones on E_{outer}. In the case of user control of an actor, the same architecture is utilized, except the user chooses what actions to take

and the same agent structure is utilized to enforce the applicable environment laws on the entity's interaction with the environment.

The Template Manager (TM), is responsible for managing the connectors for the agent and the associated tickets or frames. In the case of user control, it still manages connectors that handle cases such as collision with terrain objects and other entities. The agent architecture is summarized in Figure 66.

Actor	
View	**X3D Graphics, VRML97**
Model	**Provides constraints on our controls (ie length, width, maximum speed, minimum speed, etc)**
Controls (either User or Agent on a per-agent basis)	
User	**Agent**



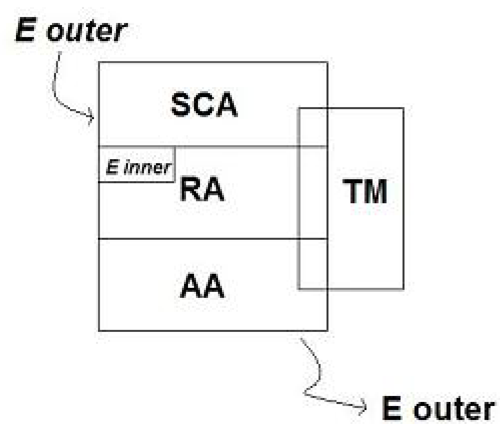


Figure 66. Depicts the agent architecture for user or agent control modes.

The Template Manager maintains a container of multiple Templates (see Figure 67). A template is defined to be a container that consists of various predefined connectors with associated vectors of possible frames of tickets to be acted upon if the connector receives a corresponding connection. The container of connectors for a template may be retracted, or added to at runtime based on the agent's performance in the environment. Additionally the frames consisting of tickets of possible actions for a given

connector may be switched, changed, modified, deleted, or added to at runtime based on the micro performance feedback of that particular sequence of actions from the environment. We can think of the dynamic creation of a new template, connector, or frame of action for a connector to be a micro implementation of the ideal of Blending discussed in [Hiles 2003], [Fauconnier 2001] with much room for future research in this area.

The template manager maintains an active template which provides information for effecting actions for the agent, but also concurrently provides feedback to the non-active templates it may have so that if the currently active template is performing poorly on the macro level it may perform a context switch to an alternative way of acting in order to possibly change the potential outcome of the given situation. Thought must be given here in the structure of the experiment. On the one hand our targeted audience may want to run the simulation in a 'rule-finding' type of mode, where they try to groom or allow the agents to discover alternative and better templates of perception in which to act, but could become frustrated if starting from scratch with templates that might visually make no sense when the scenario runs. So, running the simulation in rule-finding mode previous to deployment is essential in order to provide a starting part for the agent system.

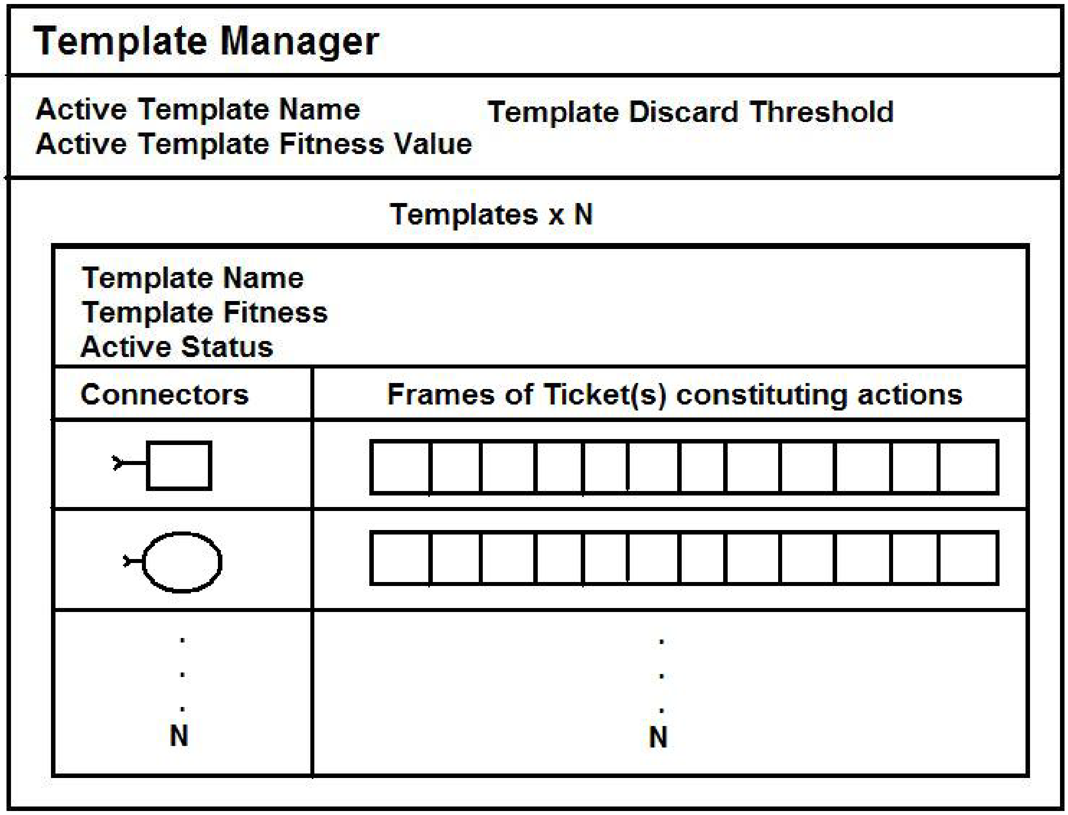


Figure 68. Depicts the Template Manager Structure.

The actions that comprise a ticket consist of basic tokens that can be combined to represent different forms of action for the agents. They consist of items such as TURN, SPEED, ATTACK, INTERCEPT, and associated values for each. A frame can consist of multiple combinations of these actions.

The process of creating new connectors with corresponding frames is similar to that of creation of a new template. If a current connector and frame, or a template, are reaching a low threshold from feedback, the Template Manager implements a low level form of blending where it treats the current pool of templates. If connectors are shared amongst the current templates, then they can be placed in the newly created one either separately or combined with the same rules applying to frames.

The Reactive Agent is responsible for providing feedback on the performance the given active template and the Template Manager coordinates the lower level feedback of

individual frames within the template structure. In the case of an Actor that is in user-control mode, there is no dynamic creation of frames or templates since the agent structure is being utilized solely for the enforcement of the laws we have implemented for our environment.

F. SUMMARY

In summary, the agent architecture implemented for this thesis has been presented and reviewed. Areas for future research in this arena include but are not limited to:

1. Extension of the current model to include multiple threats and defending craft and the investigation into communication requirements and impact on potential insights for the experimenter.
2. Further exploration of the idea of Blending as presented in [Hiles 2003], but with application towards the tactical level of war and the impact of human performance. [Wellbrink 2003]
3. Exploration into the utilization of tactical level agent model layers into an agent driven Operational Level of War model to deal with the Asymmetric threat against naval shipping.
4. Investigation into how to leverage traditional Operations Analysis techniques with emerging agent based technologies to best provide warfighters with tactical and operational level analysis tools for use in the field.

IX. EXEMPLAR USE CASES

A. INTRODUCTION

This chapter first presents different application-deployment options and methodologies investigated during the course of this research. An in-depth use-case of how one would configure and run a scenario instance is then examined from start to finish providing in-depth explanation of the various steps.

B. APPLICATION DEPLOYMENT AND UPDATES

Although the Java programming language was developed and deployed to be Operating System independent, the ability to enforce specific Java versions on the targeted client machine remains problematic without examining and deploying an installation strategy. To this end, the following technologies were investigated in an effort to find a solution for the targeted end-users of this work:

1. Heavy-weight Client-Side Applications

The ideal of a heavy-weight client application [JNLPSPEC 2000] means that we force the end-user to spend time on the first installation of our application, but that on successive application runs the user waits little to no time for running the application. Two web-based technologies investigated to this end were Sun Microsystem's JNLP and Web-Start and ZeroG.com's applet based installation-creator program, Install Anywhere.

1.0 JNLP and Web-Start

JNLP and Web-Start have a great deal of potential for the future of web-deployment of Java technologies. The JNLP file is an XML based delivery file referenced within the HTML or XHTML of the web-page a client has opened. It contains references to such things as the required version of Java, version dependencies, the code-base to download, etc which is then downloaded and launched in the Web-Start browser in a separate security sand-box from the traditional web browser sand-box. The first time that an application is launched in this manner it is downloaded in its entirety, with successive runs doing a timestamp check with the downloaded code-base's URL for incremental updating of required resources. An example invocation of the Java Web-

Start application manager desktop is shown in Figure 69.

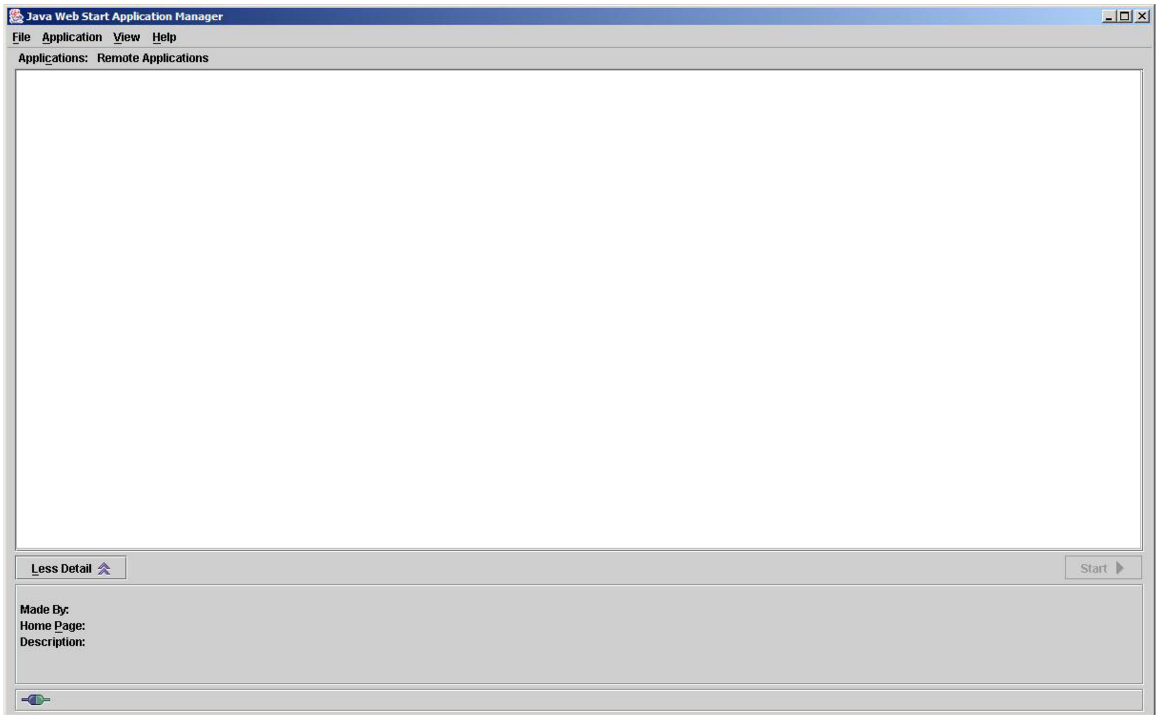


Figure 69. Example of the invocation of the Java Web-Start Application Manager desktop.

2.0 Applet-Based Installation same

The next deployment option investigated (and ultimately settled up on) was the ZeroG.com commercial Java based installation creation application Install Anywhere(Figure 70). Ease of use and quality of finished product were the primary motivators in this selection. The application allows one to set complicated application classpath and path settings in a what-you-see-is-what-you-get (WYSIWYG) manner, make a native binary executable to invoke the Java application with a privately built and crafted Java Runtime Environment (JRE) allowing the client computer to maintain its own version of the Java programming language run time environment separate for each application that relies on a specific version or library. Ease of installation for end users is excellent, reliable, and completely thorough. A typical installation configuration screen is shown in Figure 71.

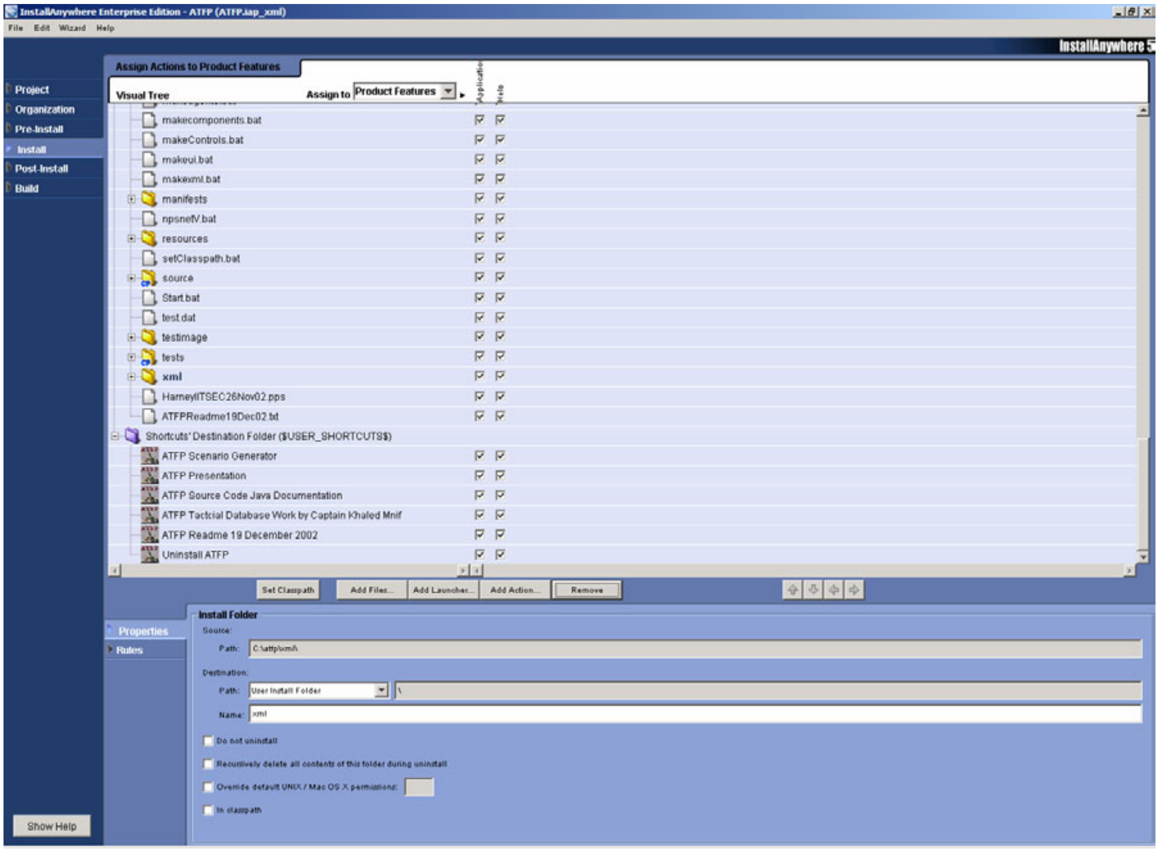


Figure 70. InstallAnywhere from ZeroG.com installation action configuration screen for creating the ATFP Scenario Generator application installation.

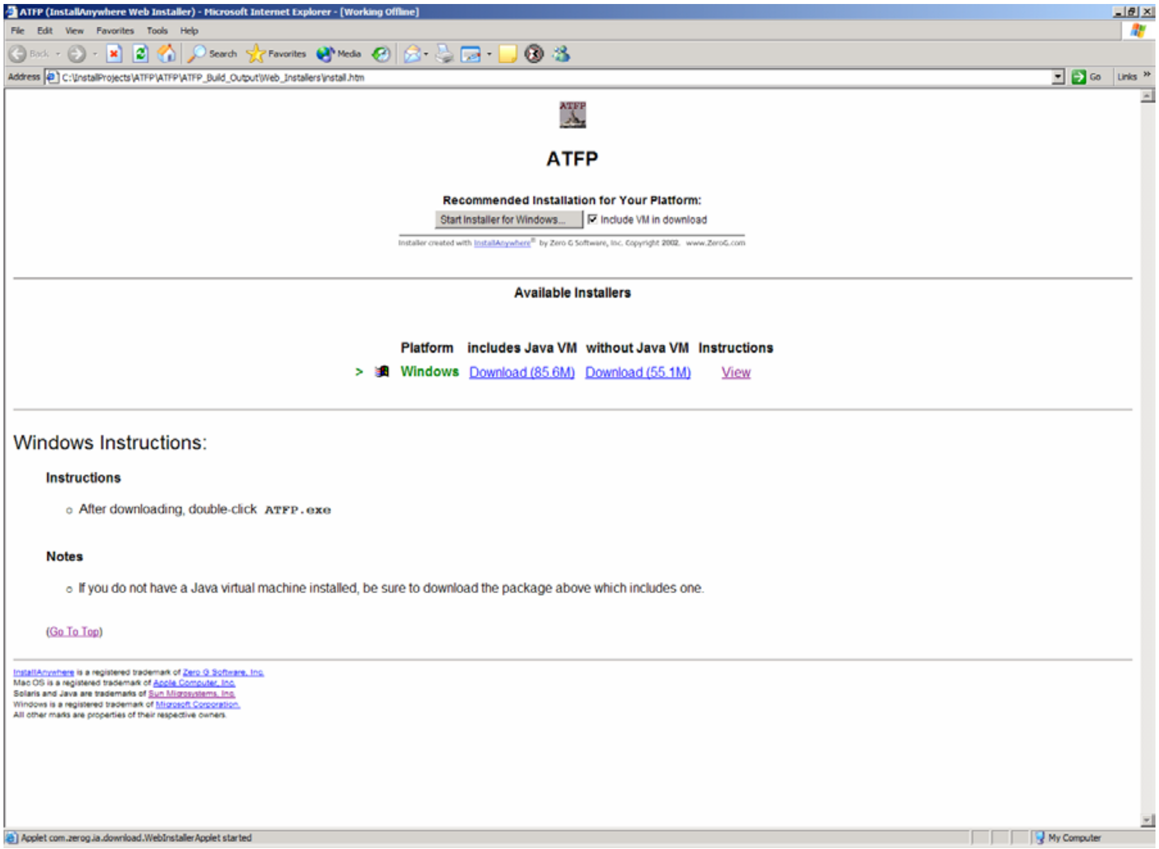


Figure 71. Resulting installation Java Applet for the AT/FP Scenario Generator application.

C. SCENARIO CREATION

Similar to what was developed during the initial usability study, iteration on the same design process continued for adding greater functionality during the remaining development of this thesis and making corrections for shortcomings as they have made sense to do so. First, the steps required to create an AT/FP scenario to the point of being ready to view it are covered.

When the end-user first starts the application they are presented with the screens shown in Figures 71 and 72.

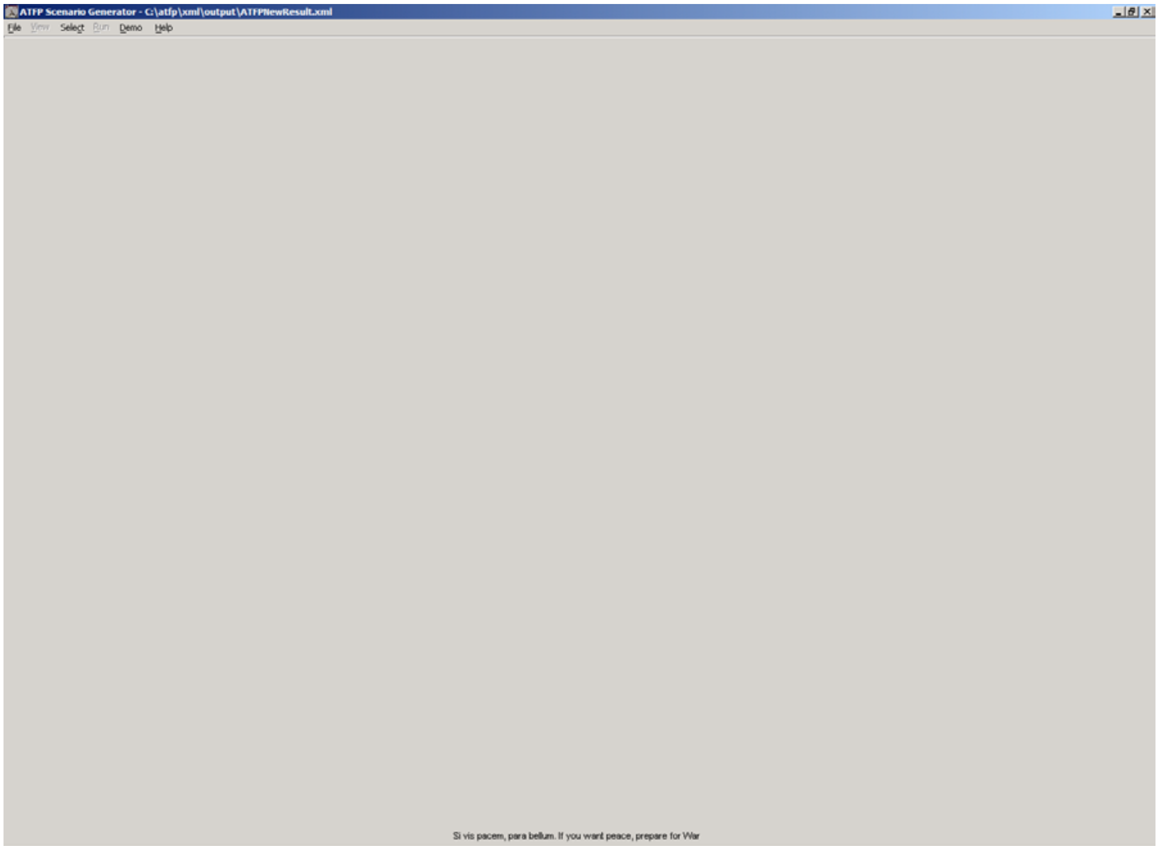


Figure 72. Depicts the main application content display for the ATFP Scenario Generator Application.

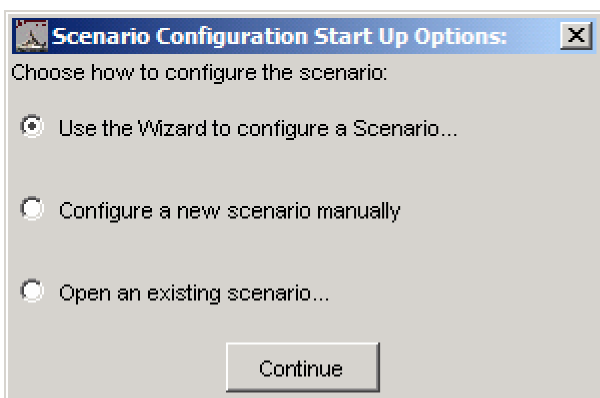


Figure 73. Depicts the Scenario Configuration Startup options the user can select from to initiate an application session.

We will first review the manual configuration modes the user must step through.

The next step is to select a harbor from the available listing of ports in the menu selection (Figure 74).

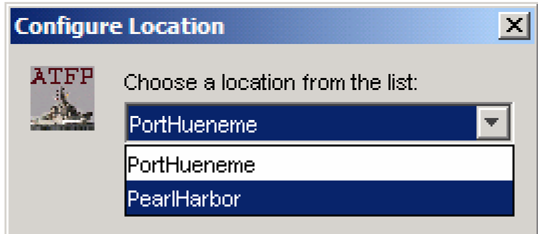


Figure 74. Depicts the available menu selection for choosing a harbor for a scenario run.

For this example we will walk through a short scenario setup that takes place in Pearl Harbor, Hawaii. The next screen presented to the end-user is a 2D chart or overhead image of the selected location to give a rough feel for the area being configured. We can't assume the end-user has been to the port being planned for in this defensive setup, but can consider it a good possibility, so we just include the chart and/or imagery for this purpose as depicted in Figure 73.

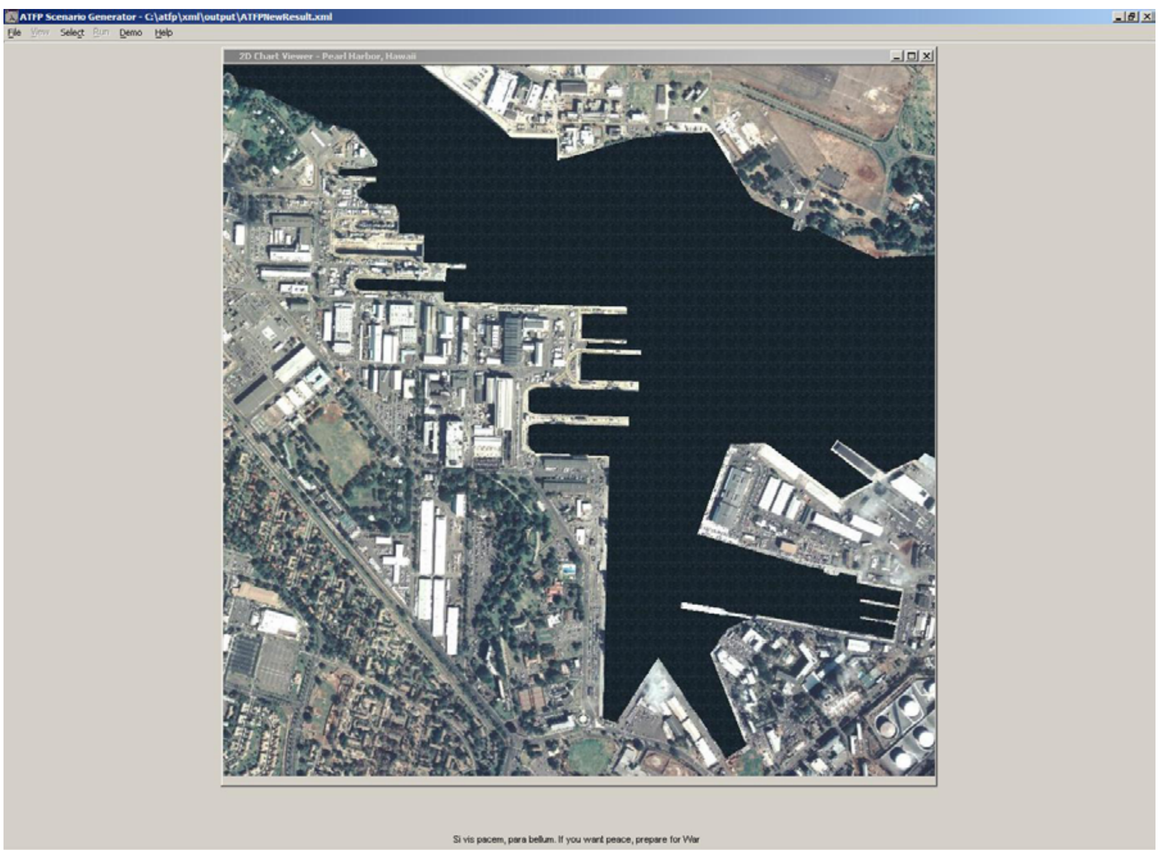


Figure 75. AT/FP Scenario Generator after the user has chosen Pearl Harbor, Hawaii as the location for the planning.

The user's next step is to configure the frequency of background shipping for the given port or area through a simplistic radio button assignment and selection process (Figure 76).

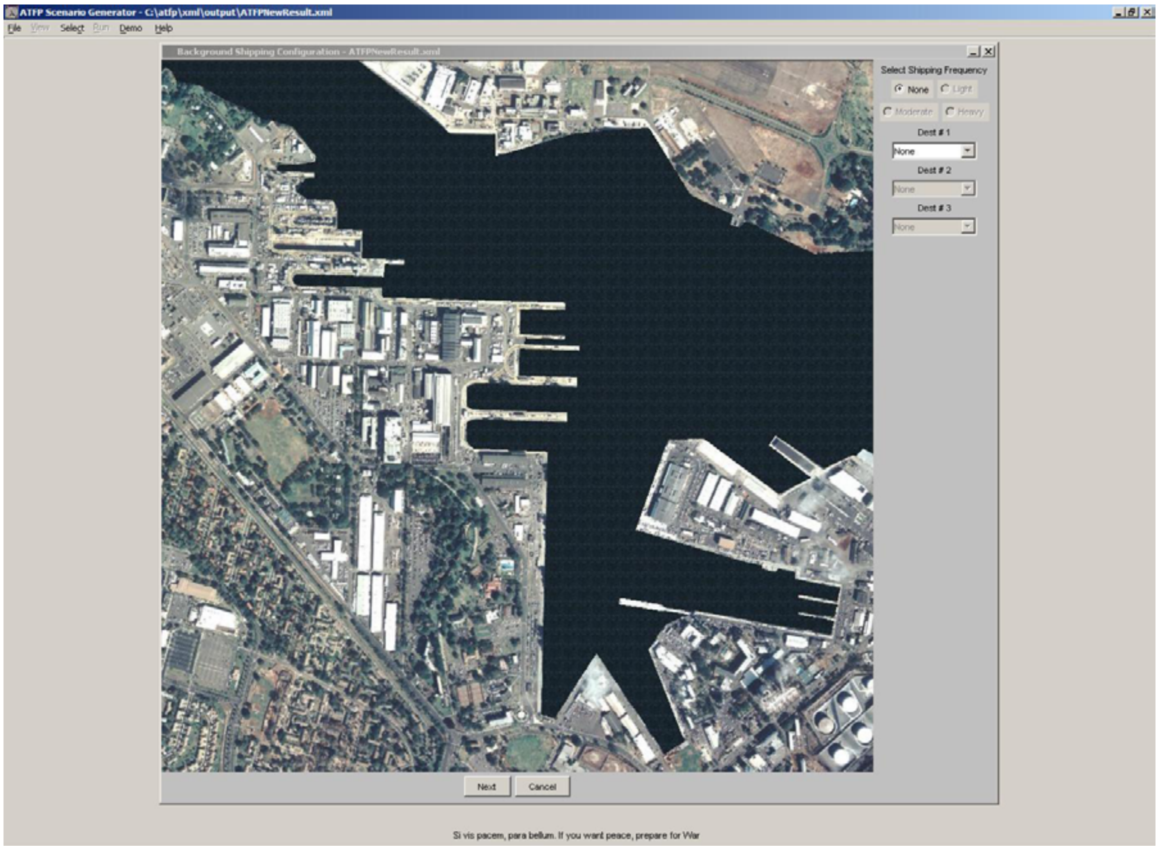


Figure 76. Depicts background shipping frequency configuration.

The next step for the end-user is to select a high value unit (HVU) to model and further configure the defensive setup he or she will have to protect this ship (Figure 77). These configurations consist of manipulation of the ranges of defensive parameters and configuring further model and control properties for the various entities within the scenario.

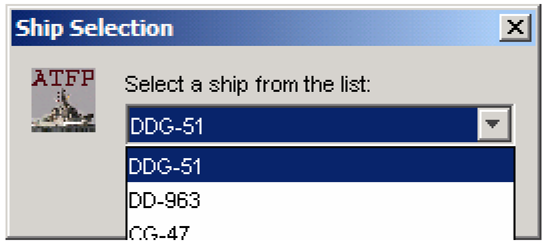


Figure 77. Depicts available ships for the AT/FP application.

Once the user has selected a high value unit, specific information on the platform is displayed as a web page from cached unclassified information from the Federation of American Scientists (FAS.org) web site (Figure 78). If the client machine is on an active network, then this information is displayed live and hyperlinks are selectable for further information searching to aid in any training needs the user may have.

Figure 78. Ship Information Screen that is displayed after the user selection.

Following this selection, the user configures the defensive setup for the current scenario (Figure 79).

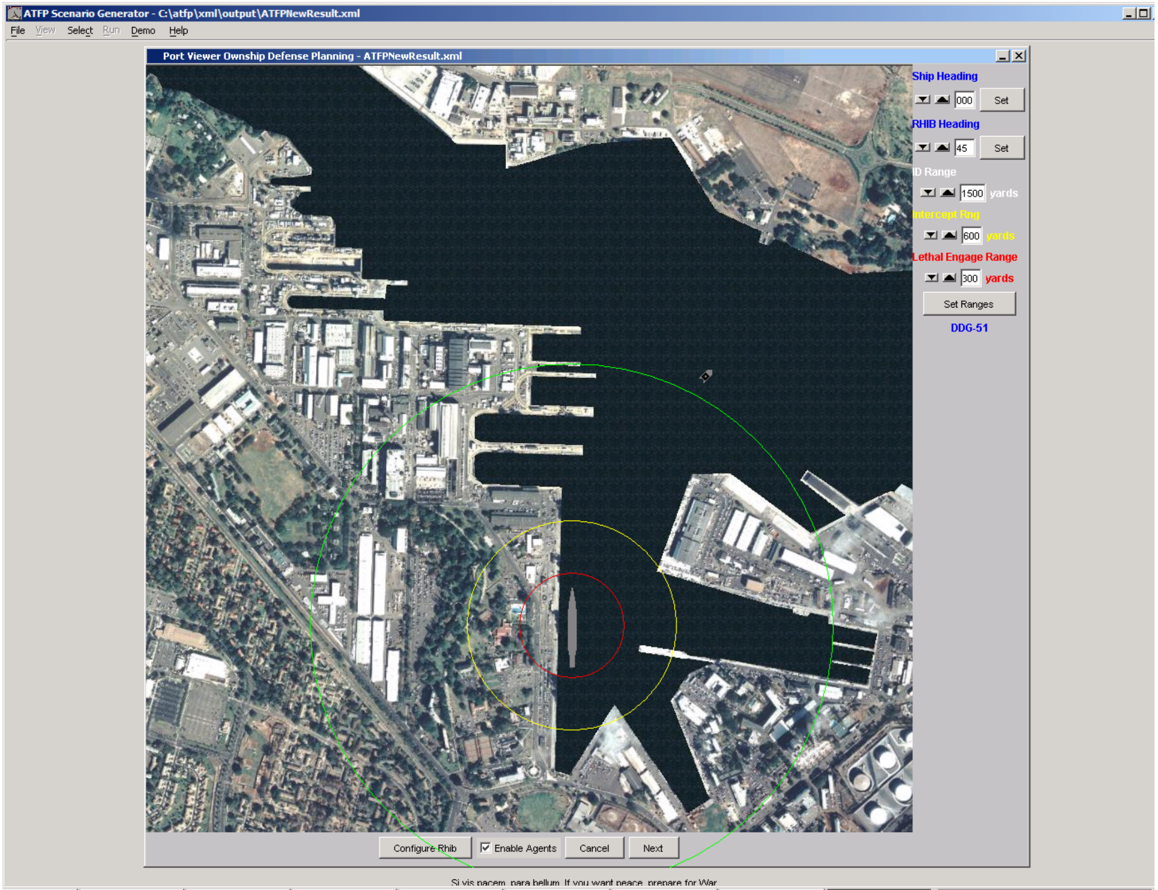


Figure 79. Defensive Setup configuration for the ATFP Scenario Generator.

The defense setup is focused on defense against the surface-borne terrorist threat and consists of placement of the HVU, placement of a defending picket boat, and configuration of tactical range parameters for identification, intercept, and lethal engagement of approaching craft. Additionally, the user has the option to select agent control or user control in addition to setting picket boat model parameters for use in the scenario as depicted in Figure 80.

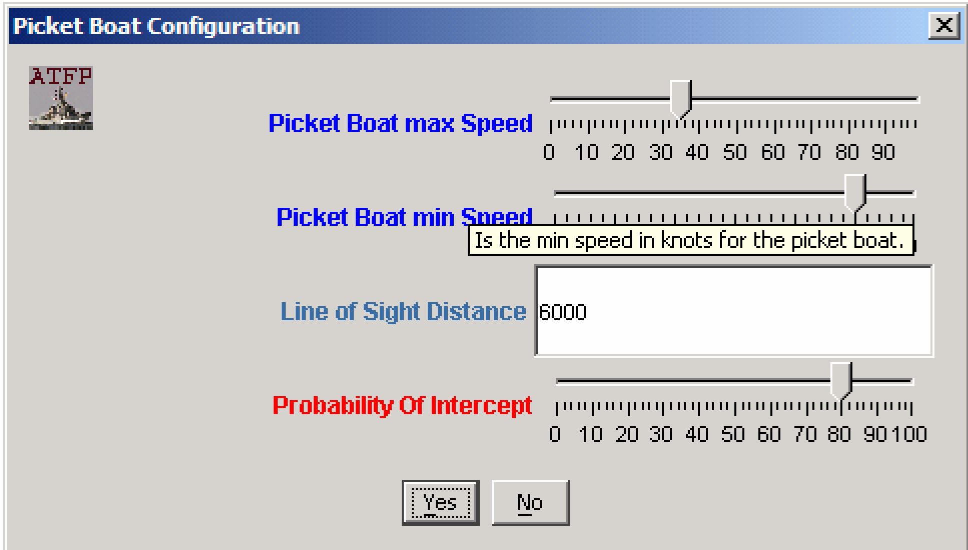


Figure 80. Depicts available options for the configuration of picket boat model parameters.

Once these steps are completed, the user can now configure the terrorist boat threat to run against the defensive arrangement. Done in a similar manner, the user is able to place the terrorist boat and select agent or user mode with similar model parameter exposure available as for that of the picket boat (Figures 81 and 82).

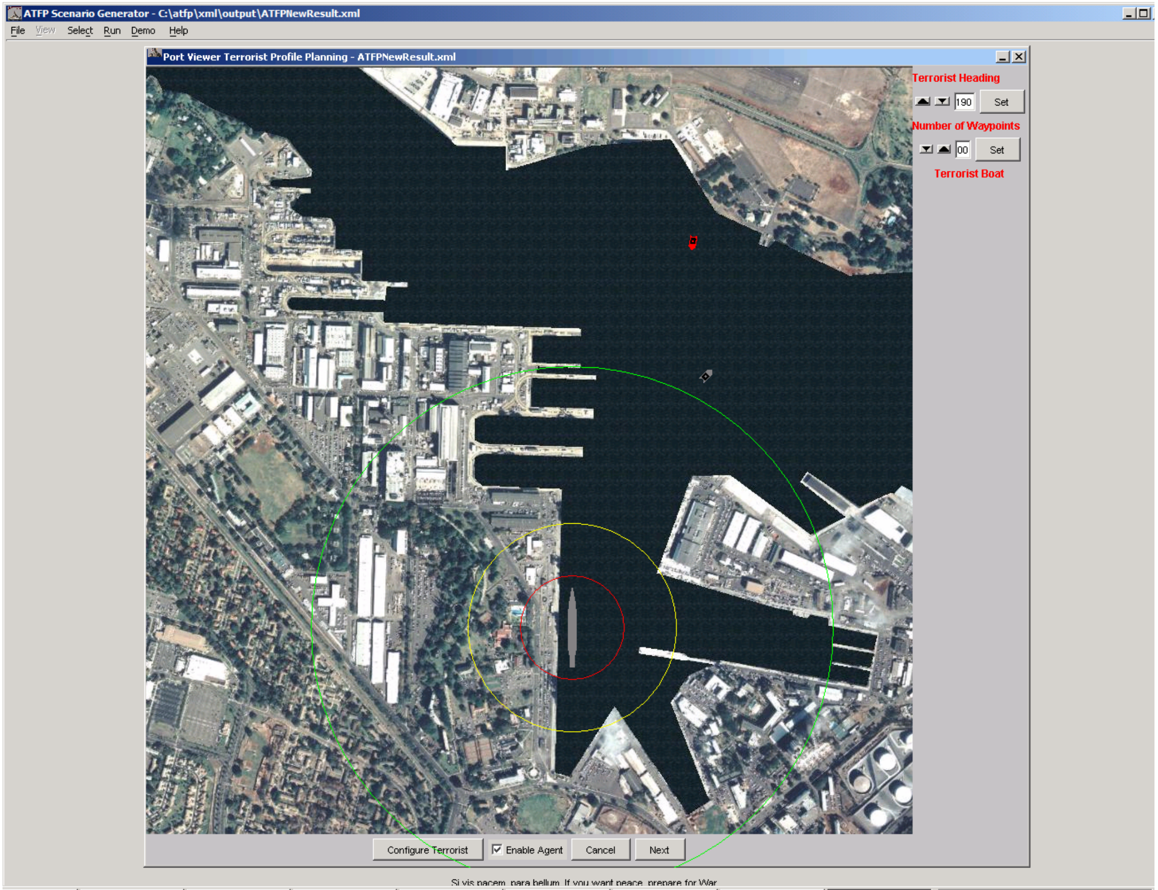


Figure 81. Depicts the terrorist boat attack profile setup.

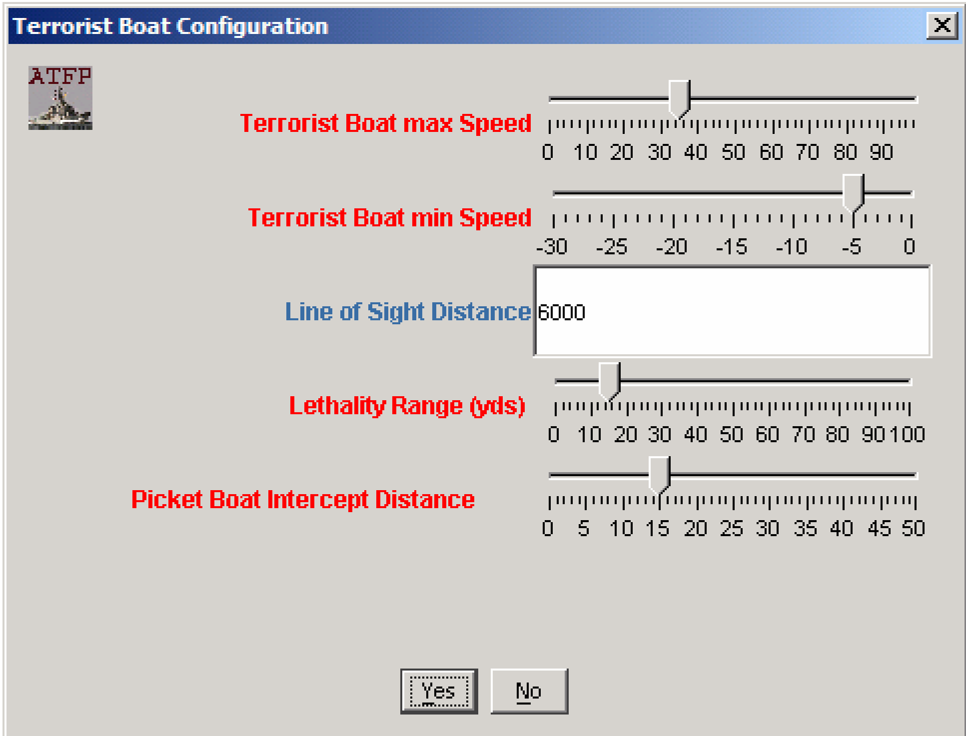


Figure 82. Depicts Terrorist Boat model parameter configuration panel.

The scenario is now fully configured and able to be run visually or off-screen for statistics depending on the end-user's needs.

D. SCENARIO VIEWING OPTIONS

At this point, the user has a few options available to for executing the configured scenario. The user can view in 3D to gain visual insight to possible defensive plan shortcomings, experience emerging tactical situations from the surface perspective of protagonist / antagonist, or execute non-rendering runs for statistical data collection.

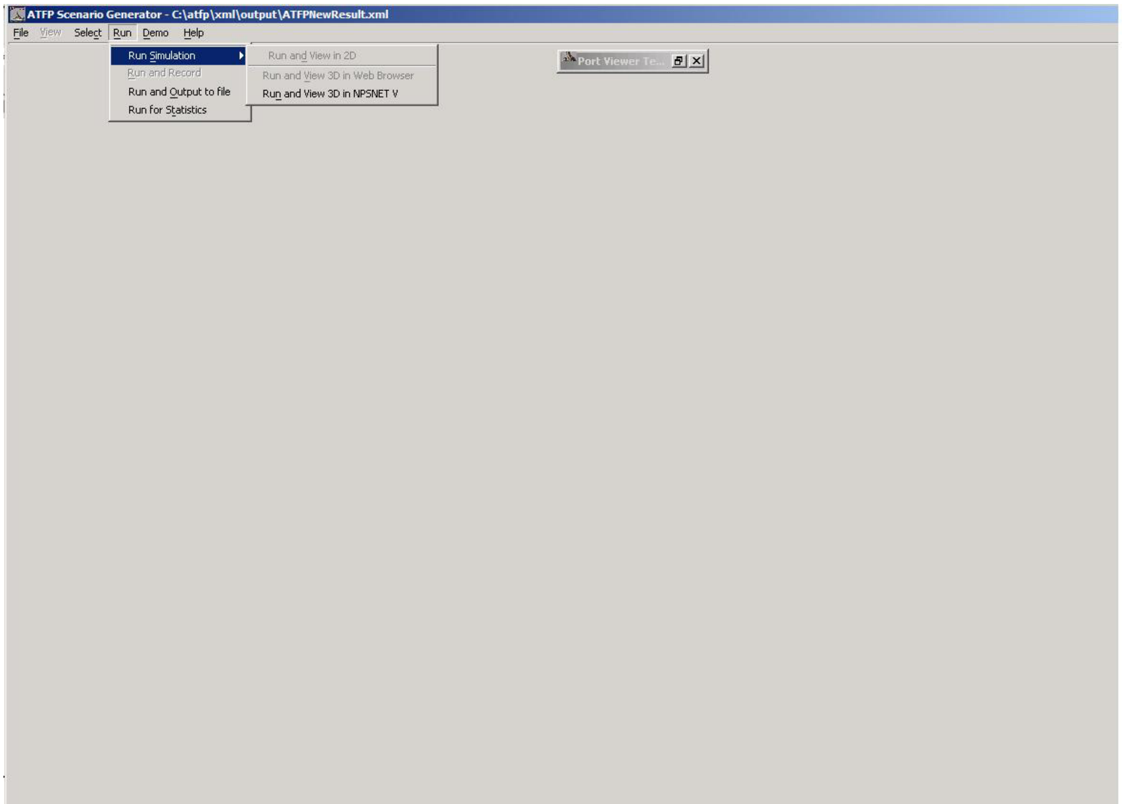


Figure 83. Depicts available menu options for running the scenario once configuration is complete.

1. 3D View

When the 3D View option is selected, the application creates the 3D scenario dynamically by applying the applicable XML stylesheet to the current scenario instance document. The scenario instance document is serialized to disk if the user has not saved the application with the default filename ATFPNewResult.xml. The 3D scenario is quickly created, then loaded into the application desktop. Once loaded, the agent and (if applicable) user controls are started and the scenario can begin (Figure 84).

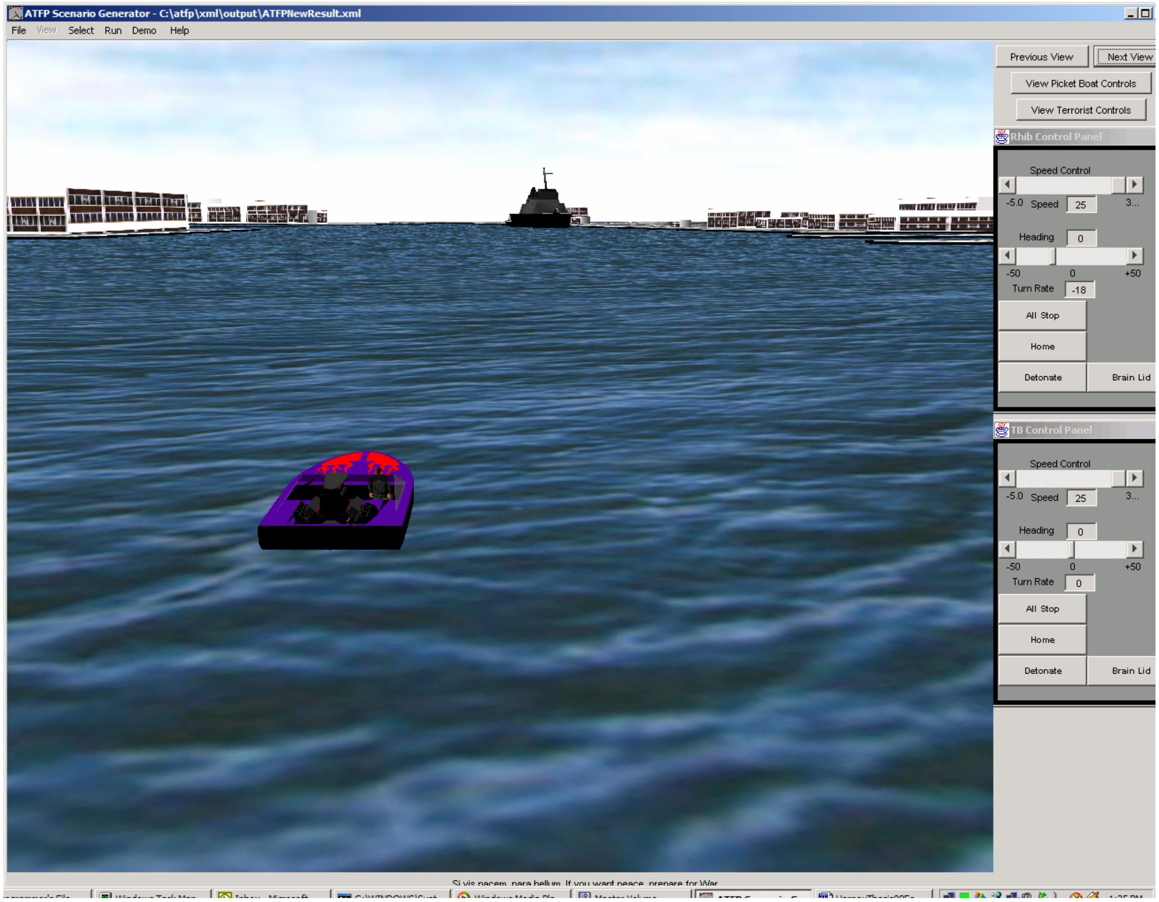


Figure 84. Depicts the 3D view of the scenario in action from the rear perspective of the hostile agent-driven terrorist boat.

When a visual run is completed, the scenario pauses for a few moments, then displays the single run results to the end-user in case they had their attention diverted and did not see the outcome of the attack.

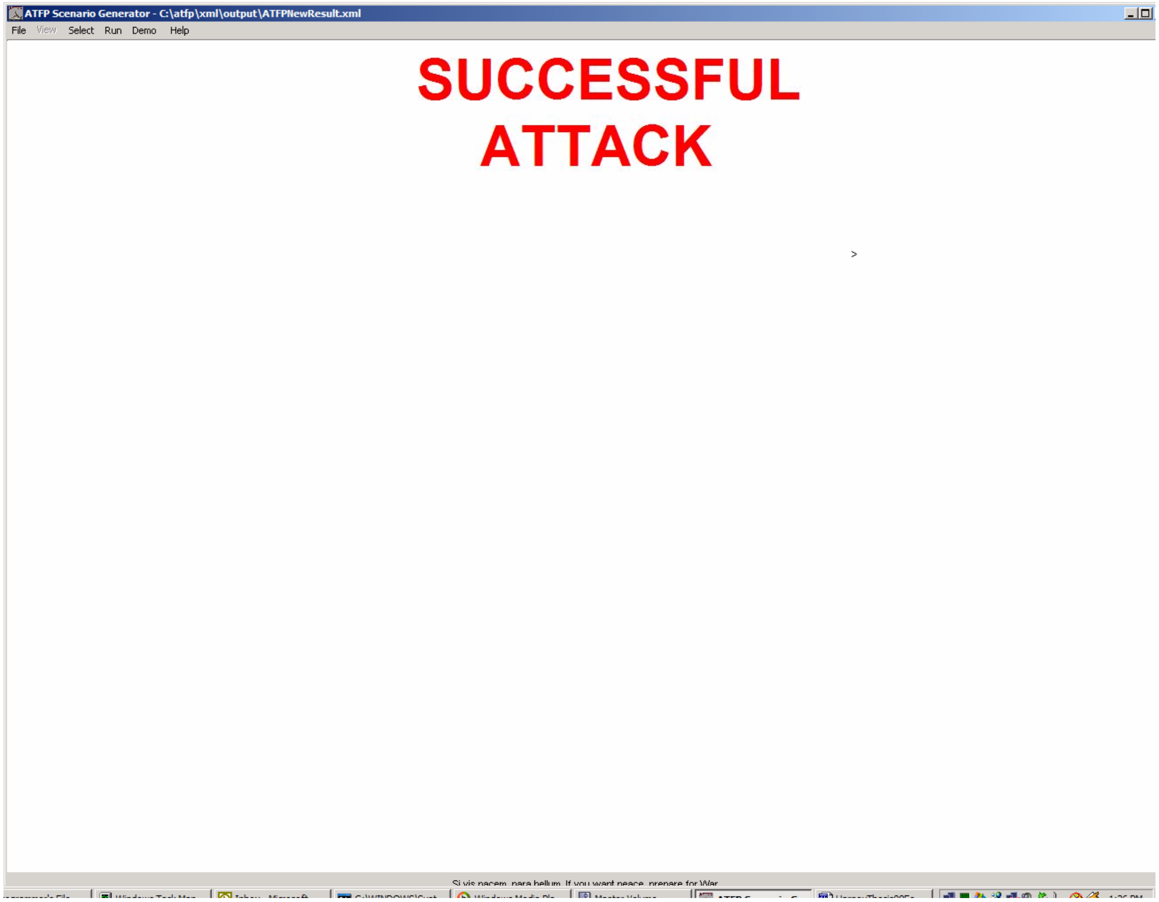


Figure 85. Depicts one run results for an AT/FP scenario run.

The user is then presented with three options as shown in Figure 86: 1) View results of another scenario run , 2) run for statistics, or 3) start a new scenario. Additional single-run results consist of displaying tactical data utilizing an XML stylesheet from [MNIF 2003], in addition to a second XHTML page that depicts scenario setup information visually in the form of screen snapshots taken while the user was in the configuration phase of the scenario setup.

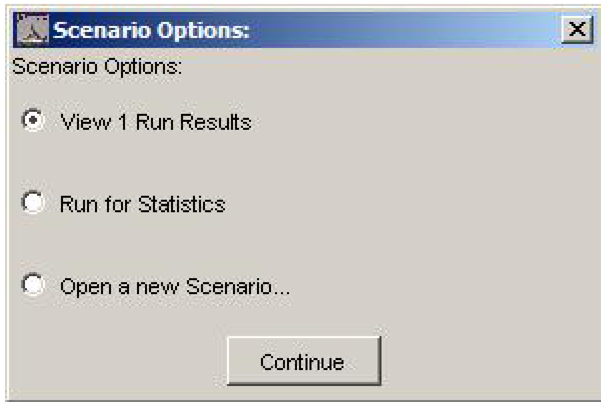


Figure 86. Scenario options presented to the end user after a single-scenario run.

2. Non-rendering Scenario Runs with Statistics

When run without rendering, the user is able to iterate over multiple scenarios in less time than visually running multiple scenarios. However, since the simulations are using a real-time physics based scenario model there is a limit to the speed-up possible. Further optimization is possible as future work. An example of the SVG results of multiple runs for statistics is shown in Figure 87.

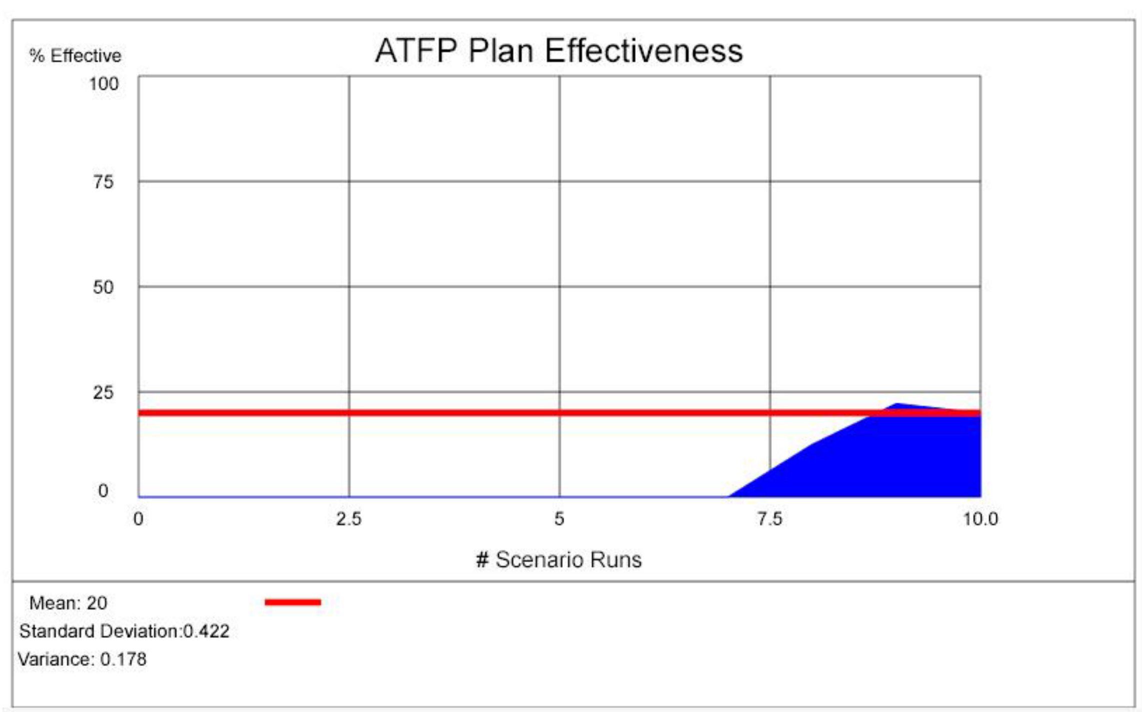


Figure 87. Statistical output from ten scenario runs rendered in SVG graphics.

E. USING PREVIOUSLY CONFIGURED SCENARIOS

The ability to save scenario files to disk and reload at a later time became an essential feature once the number of locations and ship types supported in the application grew. For this reason, the end-user is able to do both.

To load and view from disk, the user can either choose to open a scenario file from the wizard menu or from the file menu using a standard file-chooser manner common to most modern windowing applications (Figure 88). If the file chosen is not a valid AT/FP Scenario Instance file, then an error message will be displayed to the user and retry is permitted. If the file loads satisfactorily, then the defensive scenario configuration screen is automatically loaded. If no changes are desired, the user simply has to select the 'NEXT' button at the bottom of the defense and terrorist configuration screens in order to proceed to executing a visual or non-rendering run for statistics.

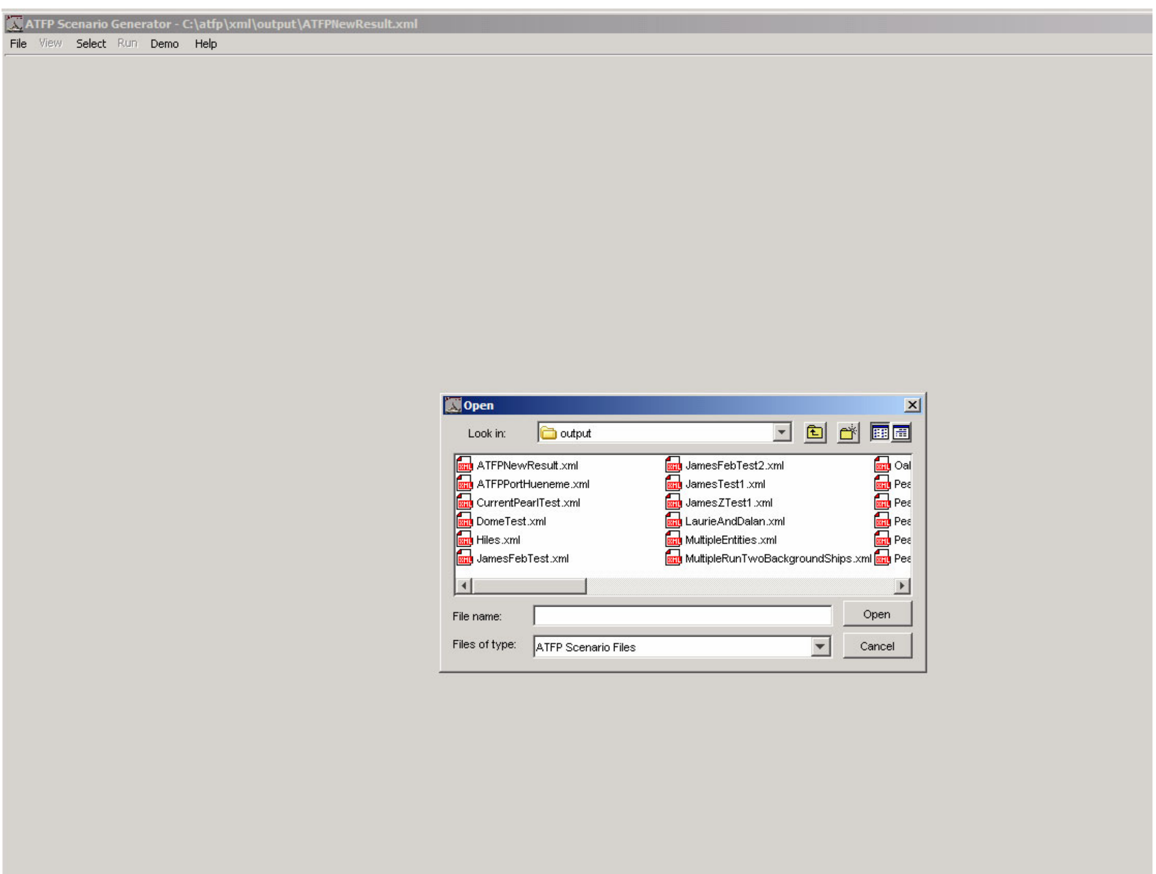


Figure 88. Depicts the File Chooser dialog for opening a scenario file from disk.

F. SUMMARY

This chapter describes scenario configuration, viewing, and loading features of the AT/FP scenario configuration application. Additionally, deployment strategies investigated and developed during the development of this application are presented.

X. APPLICATION TOWARDS U.S. NAVY TRAINING, EDUCATION AND EXPERIMENATION

A. INTRODUCTION

During the course of this thesis, we partnered with colleagues from the Naval Postgraduate School's Wayne E. Meyer Institute of Systems Engineering to take advantage of several research opportunities that would not have been possible otherwise. Namely, we were able to explore modeling and simulation work for visualizing and playing 'what-if' scenarios in the context of Commander Third Fleet Limited Objective Experiments focused on Anti-Terrorism and Force Protection. We also gained exposure to current doctrine methodologies being examined for implementation and dissemination through the U.S. Navy's training and education system.

B. LIMITED OBJECTIVE EXPERIMENTS

Two of the Wayne E. Meyer Institute's primary goals are : 1) to integrate and enhance existing systems engineering programs at the Naval Postgraduate School and 2) to provide unique graduate education opportunities through experimental design, coordination, and execution. This proved to be the case early in the course of this thesis research by being afforded the opportunity to create a real time simulation of implementation of experimental non-lethal weapons technologies being examined for use in a live experiment at Naval Base Port Hueneme, California (Figures 89 and 90).

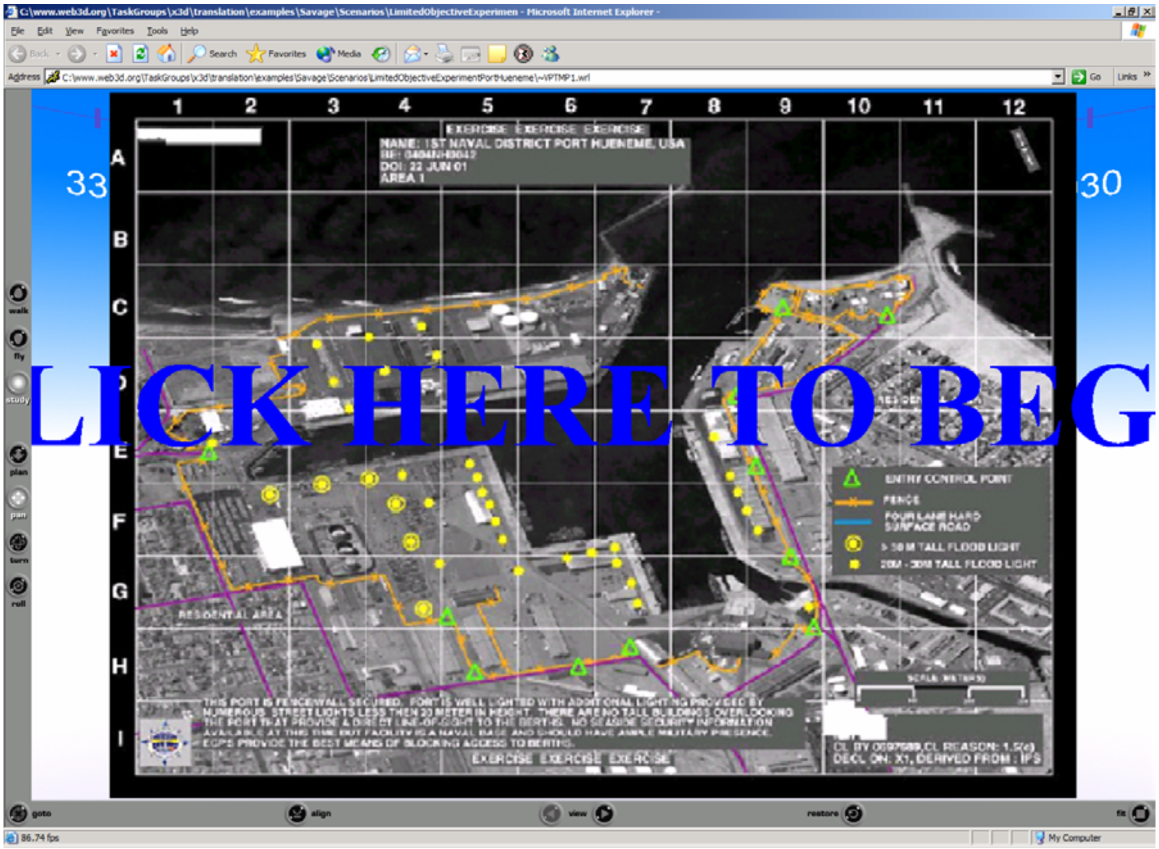


Figure 89. Depicts the entry level view for the LOE real time simulation depicting 2D imagery with exercise tactical data prior to viewing the 3D scenario.

Research and development in a pure web context utilizing basic agent technologies to assist the analysis of the limited objective experiment (LOE) execution prior to the experiment taking place was one of the primary motivators for the initiation of this phase of the thesis research. The LOE planners requested development of visualizations of the primary scenario events and the proposed area of operations to aid with their pre-experiment planning and post event reconstructions. [Blais 2002B] Of particular note: **while creating a preconstruction of possible scenario events or conducting a static reconstruction following the experiment's conclusion can prove interesting, being able to play 'what-if's' to gain greater insight towards execution of future exercises or real events proves to be one of the true powers of modeling and simulation (M&S) technologies.**

Therefore, agent-based technologies were incorporated early in the thesis work to investigate a greater spectrum of potential outcomes to enable an end-user or analyst to

possibly gain greater insights (Figures 91 and 92). As a result, the web-based virtual environment becomes an experimental laboratory supporting investigation of what really occurred and the manipulation of model parameters and other factors to see what could occur.

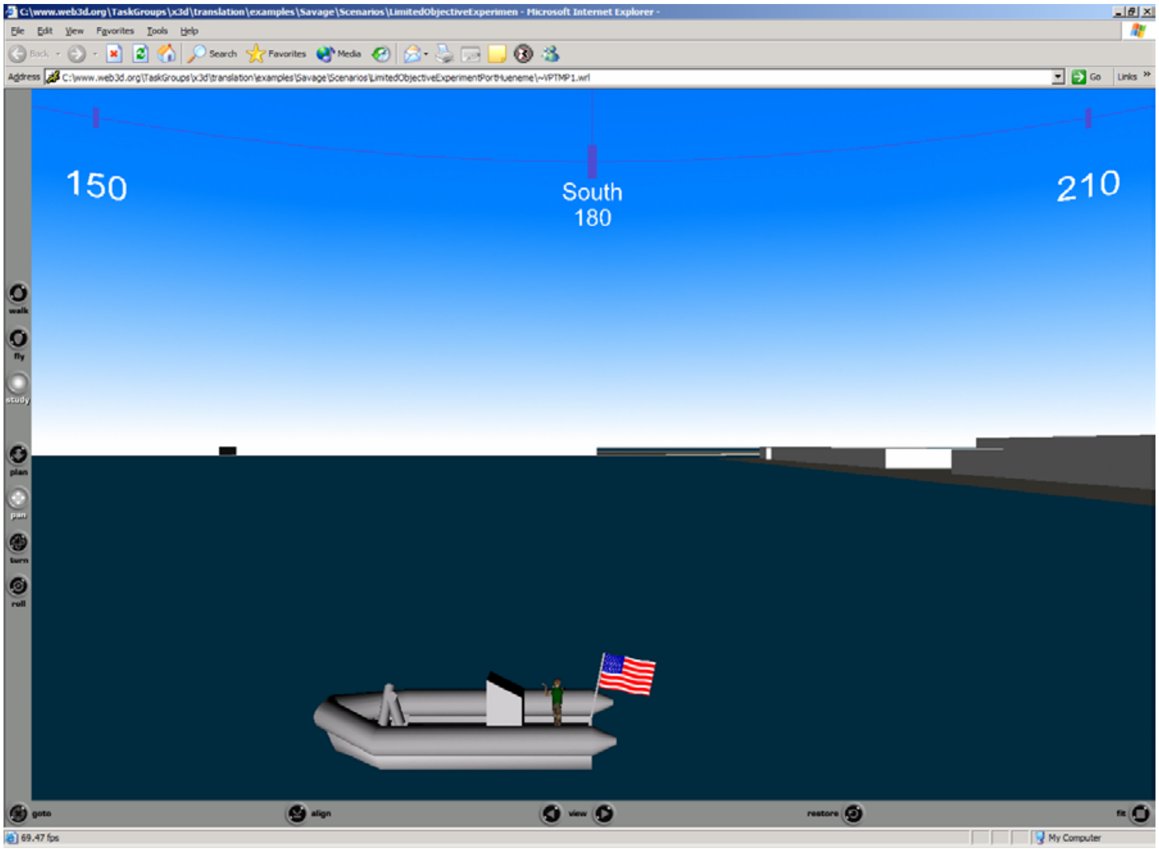


Figure 90. Depicts the conceptual view of a non-lethal net engagement system being modeling prior to use onboard a U.S. Navy rigid hull inflatable boat(RHIB).

For the LOE, the use of multi-layered agent-based technologies allowed the analyst to leverage the simulation in several manners. First, was to gain insight on the employment possibilities of a non-lethal net entanglement system to be deployed from a defending RHIB boat against incoming threat craft. Specific items of interest to the experimenters for this capability were: 1) the ability to run various threat profiles against pre-configured defensive postures in order to make the most effective use of real-world experiment assets, 2) the capability to try different variations of defensive behaviors to gain insight towards the most effective employment means of the entanglement net, and 3) the ability to experiment with AT/FP doctrine under development for handling the

detection, sensing, decision, and engagement components of the tactical employment of defenses.

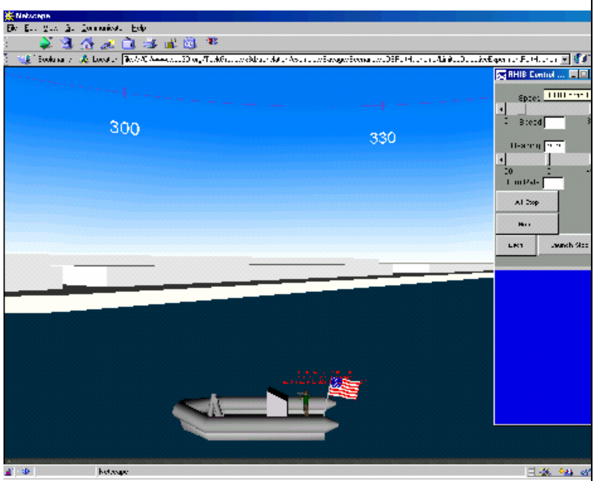


Figure 91. Depicts a defending RHIB boat in user-control mode for the LOE simulation run.

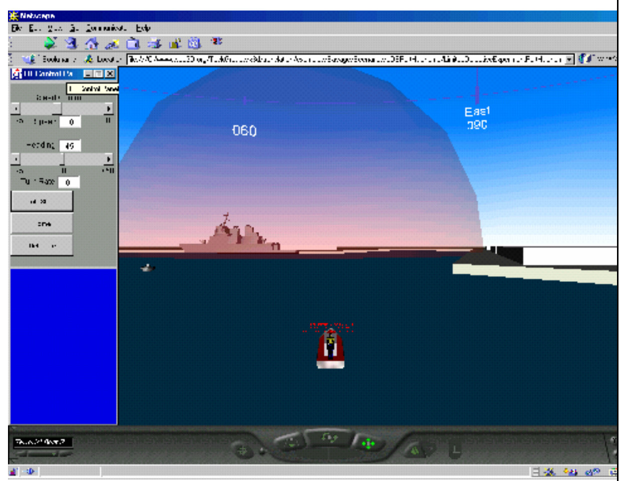


Figure 92. Depicts the graphical representation of an attacking surface craft in agent-control mode. The red sphere depicted is indicative of the high value unit's lethal engagement range for this scenario run.

Next, when setting up the experiment, there was a need for a means to gain insight for where shipping exclusion zones need to be placed with respect to the mooring location of high value units in the harbor. With the ability to dynamically define the experiment setup, initiate a scenario run, and view possible threat axes that would place the defensive units at a disadvantage, one could either use this information for better defensive setup or to better configure an attack to test shipboard defenses vice picket boat defenses.

Also, the effect of decision-making time on the execution of defensive doctrine can be examined with the impact on the probability of success of various defensive configurations. Additionally, the experimenter could decide to iterate over multiple scenarios with the same input parameters for the various models to see if statistical insight can be gained for the given set of inputs. Some questions that were looked at in this manner were insights to tactical parameters such as the optimal picket boat placement against the surface threat, effective ranges to configure tactical parameters based on the harbor geometry configuration, and insight into placement of defensive

layers in order to best stymie a terrorist attack. In this manner, the analyst can have this information available to decide how to best arrange the forces participating in the LOE and avoid wasting money or time unnecessarily with bad experimental runs in the field. The ability to operate with the human in the loop was also incorporated so if the analyst had a specific 'what-if' tactic that he or she wished to run they could do so without having to try and figure out how to force the agent model to replicate exactly the threat profile or behavior desired, if possible at all.

Scalable, dynamic, multi-user simulation for this research was achieved by integrating the Distributed Interactive Simulation (DIS) standard protocol [IEEE 1995], Java software, and the VRML graphics format (obtained from translation of X3D files) [Brutzman 1998]. The DIS protocol was used to communicate state information among the multiple entities either participating on a shared network or on a single computer. For the LOE simulation work, the DIS-JAVA-VRML integration was effectively used as the means to control and render the simulation based on user or agent control inputs.

C. LEVERAGING EMERGING WEB-BASED VISUALIZATION FOR TRAINING AND EDUCATION

From the LOE simulation work evolved a continuing working relationship with the Wayne E. Meyers Institute. Research continued along the lines of investigating what the actual tactical process should be for shipboard crews operating in hazardous port environments and providing feedback to the analyst on the impact of non-standard execution of strictly defined doctrine. Specifically, providing both tactical feedback such as that shown in Figures 93 and 94 after scenario runs as well as providing dynamic information in the context of scenario configuration were added to help the thesis application be leveraged as both a simulation and training tool.

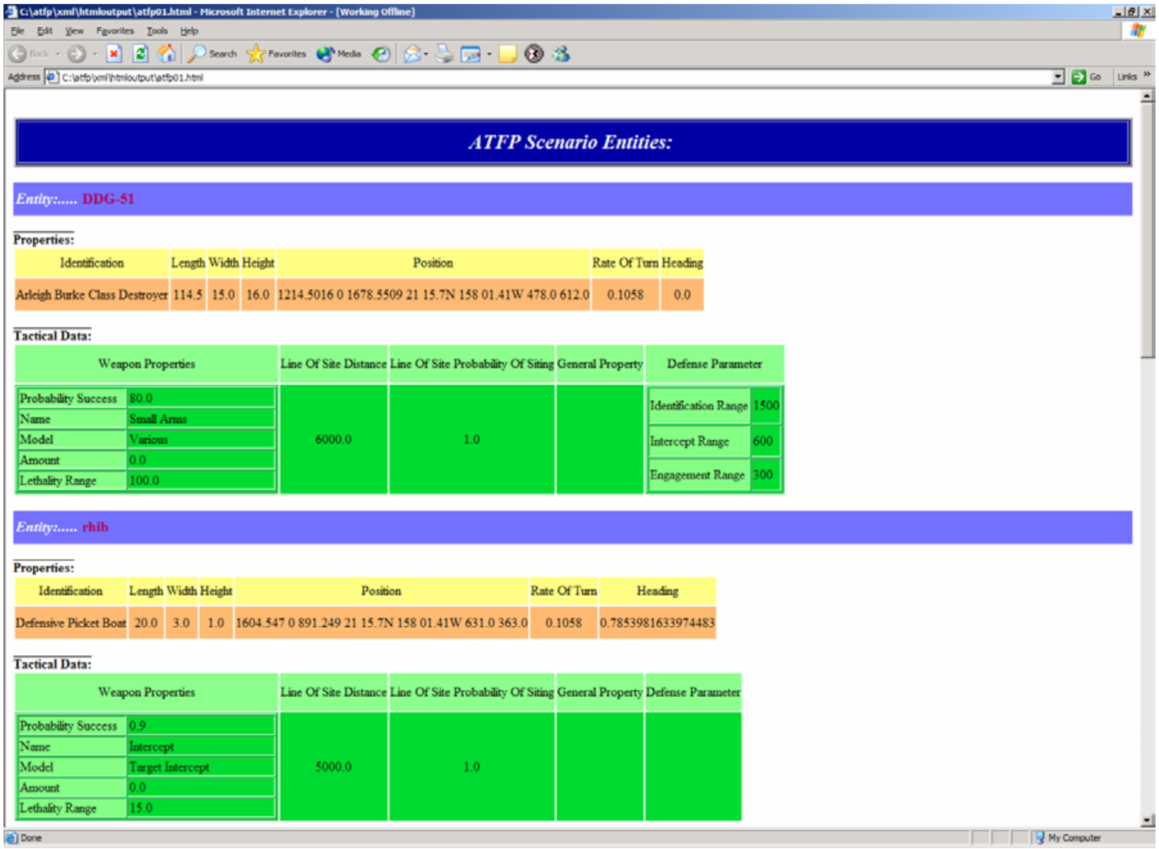


Figure 93. Depicts styled tactical data created and made available to the end-user after viewing a scenario iteration. [Mnif 2003]

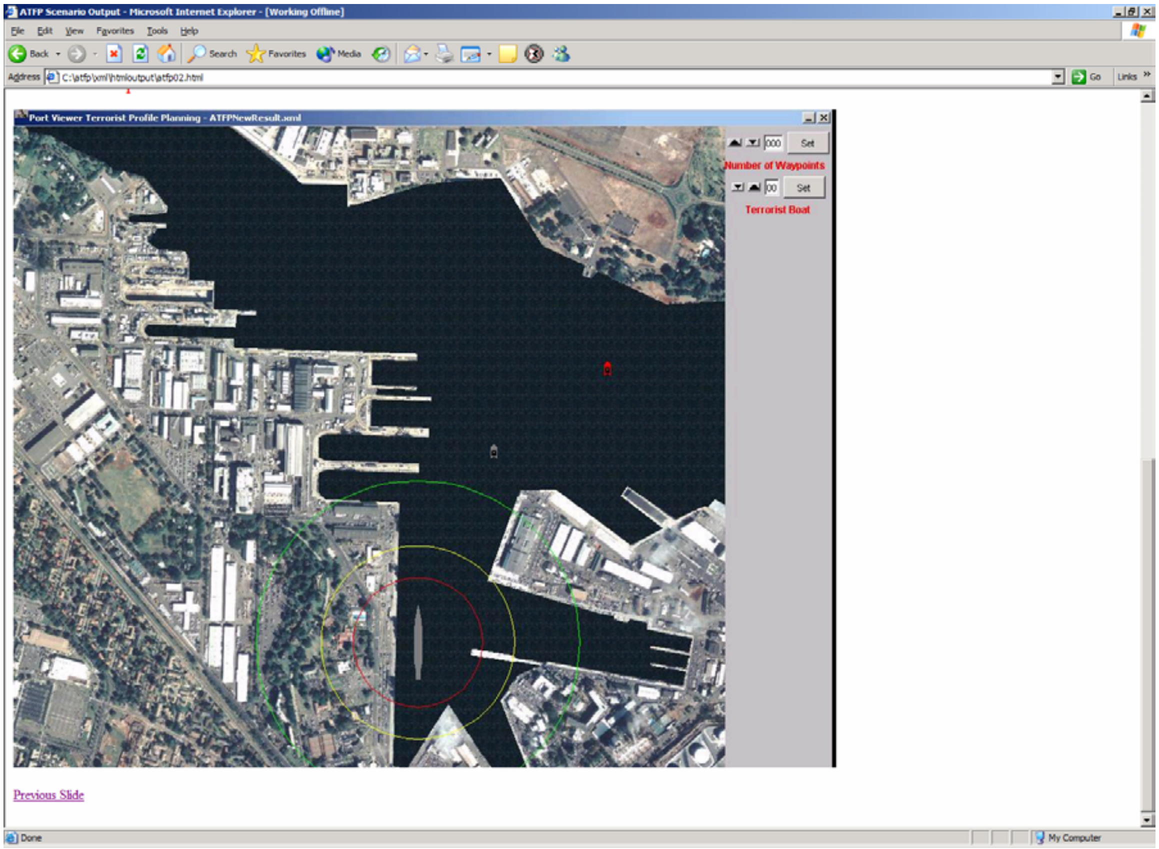


Figure 94. Depicts screen snapshot of the visual setup for the terrorist attack profile configuration to aid the end-user in evaluating the outcome of their defensive plan occurred.

As mentioned in a previous section, tactical data on the high value unit and other entities can be viewed in a standard web fashion by the end-user if they are utilizing the application in either a learning environment or as a ‘non-expert’ user with the ability to continue hyper-linking to other web-enabled information when the application is placed on an unclassified or secure network environment (e.g., Figure 95).

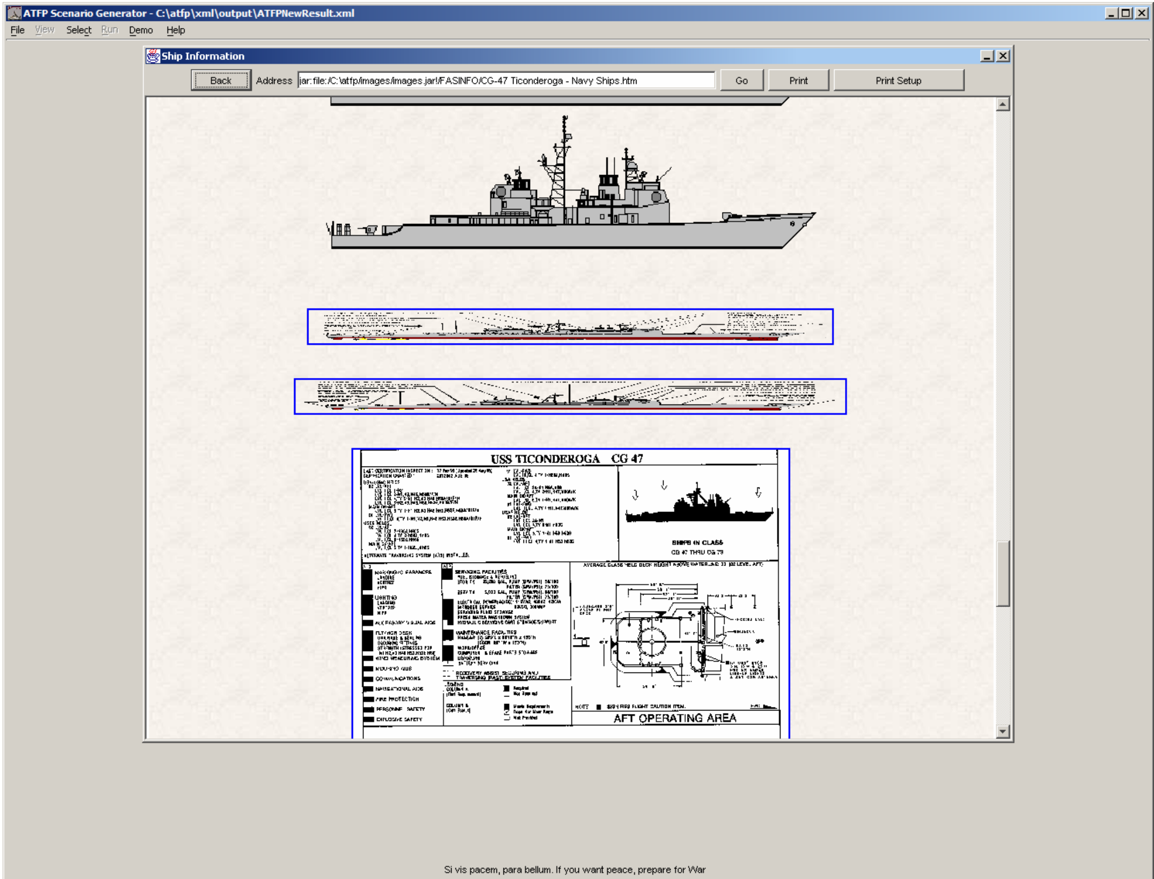


Figure 95. Depicts cached web information on the Ticonderoga class Cruiser displaying ship configuration information that might be interesting to the end-user.

D. SUMMARY

In summary, this chapter described how this thesis was leveraged in conjunction with Wayne E. Meyer Institute of Systems Engineering personnel to effect research towards application of modeling and simulation for experimentation, training and education. Current research continues for both identifying how to best leverage the web technologies utilized in this thesis within the Naval Education and Training pipeline and examining possibilities for use to aid in the development of a Concept Of Operations (CONOPS) for the U.S. Navy Spartan 'Scout' Unmanned Surface Vehicle (USV) for assured access and force protection. [Hundt 2002]

XI. CONCLUSIONS AND RECOMMENDATIONS

A. GENERAL THESIS CONCLUSIONS

The application of current and emerging web standards with agent-based technologies can be leveraged now to aid the warfighter in gaining insight for fighting current and future asymmetric threats. Using XML as a backbone data-centric architecture to base visualization and scientific modeling and simulation development proves to be both a powerful and extensible means with which to leverage IT-based efforts to bring modeling and simulation to planning tactical operations of war. This thesis contributes to current and ongoing efforts within both the Department of the Navy and the Naval Postgraduate School's MOVES Institute in the utilization of web-based M & S for defense against the ongoing threat of terrorism.

B. SPECIFIC CONCLUSIONS AND RESULTS

1. Easy, Dynamic Scene Creation

One of the primary goals of this thesis was to enable the end-user to be able to easily configure virtual environments at runtime. To achieve this, the process of: 1) defining the problem, 2) creating an example prototype scene, 3) defining an XML schema that represents the scene components, and 4) binding to an intuitive user-interface for WYSIWYG selection and configuration of scene components was utilized. Combined with the use of XML programming and other open, web based technologies, this process is both extendable and repeatable.

2. Real-Time Scene Interaction

Another goal of this thesis was once AT/FP scenes of interest could be dynamically created, was to enable them to be run-time extensible. This was achieved through the use of a socket-based architecture. Adding both agent-based and user driven controls for scenario entities that send controlling network packets from the DIS-JAVA-VRML network protocol, AT/FP simulations are able to be run in agent only mode, user-control mode, or a combination of each. This capability enhances us of the application to gain insight for defense against the water-borne terrorist threat.

3. Laboratory for Experimentation

This thesis work provides a laboratory for experimentation for both the warfighter as well as for the researcher. Typically, much time, effort, or money must be expended in order to create a suitable environment with the necessary components in to conduct specific research in artificial intelligence or human behaviors. This thesis implements an agent model in a componentized fashion so that follow-on work can augment or replace-in-full the work done in this area whilst leveraging the same graphics, model, and XML-based infrastructure. In the conduct of this work, specific relations between the scenario input parameters of background shipping frequency, harbor geometry, tactical parameter configuration, defensive picket boat placement and configuration, and threat configuration and profile were found. Specifically, as the rate of background shipping was expected to be higher, the tactical parameters relating to identification and interception ranges had to be compensated for in order to maintain a suitable effectiveness percentage for a given defensive posture.

4. Applications for Navy Training and Education

During the design and implementation of this thesis, we found that there are two areas of application of M & S tools within the current navy training and education system. First, using this work, experienced users can gain insights about the physical layout of a harbor they may not have visited before by visualizing different and varying defense scenarios against terrorist threats. These capabilities proved more valuable than traditional 2D paper chart based planning methods. Secondly, by incorporating the ability to view and dynamically web-surf or query information on the capabilities and limitations of the naval platforms that can be selected in the application it also serves as a teaching platform for junior personnel.

C. RECOMMENDATIONS FOR FUTURE WORK

1. Coordinated Development with the U.S. Navy AT/FP Schoolhouse

This research can be further extended and modified in coordination with the U.S. Navy AT/FP school house. Specifically, research can be targeted to providing immediate visual and statistical feedback to students in the context of currently developed course

modules. The focus can range from extending the capabilities of the existing model in order for the students to be able to ask more varied questions on how the different input parameters play a role in the tactical effectiveness of a defensive plan against the surface threat, to expansion of the threat areas currently modeled to include shore, air, and sub-surface threats. Additionally, specified research can be focused on measurements of a student's retention of course objectives when using this work to provide compelling feedback and training for defensive plan implementation.

2. Modification for Use with the Spartan Unmanned Surveillance Vessel

Perhaps one of the more interesting research directions lies in investigating research possibilities with the Navy autonomous surface craft, Spartan, for port defense. [Hundt 2002] Future work could include but is not limited to the agent control design for the craft to best effect a patrol and search of a harbor to which it is deployed, extension of this work to provide a control interface and feedback system for operators of the craft, and extended possibilities.

3. Continued Applied Autonomous Agent Research

Further agent research is encouraged in the following areas:

i. Extension of the current model to include multiple threats and defending craft and the investigation of communication requirements and impact on potential insights for the experimenter.

ii. Further exploration of the idea of Blending as presented in [Hiles 2003], but with application towards the tactical level of war and human performance degradation. [Wellbrink 2003]

iii. Exploration into the utilization of tactical level agent model layers into an agent driven Operational Level of War model to deal with the asymmetric threat against naval shipping.

iv. Investigation into how to leverage traditional Operations Analysis techniques with emerging agent based technologies to best provide warfighters with tactical and operational level analysis tools for use in the field.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. ACRONYMNS AND ABBREVIATIONS

AA	Action Agent
AT/FP	Anti-Terrorism / Force Protection
API	Application Programmers Interface
CA	Composite Agent
CAD	Computer-Aided Design
COMPACFLT	Commander-in-Chief Pacific Fleet
CMAS	Connector-based Multi-agent System
CONOPS	Concept of Operations
CPDU	DIS Collision Protocol Data Unit
CPU	Central Processing Unit
CTF	Capture The Flag
DES	Discrete Event Simulation
DIS	Distributed Interactive Simulation, IEEE network protocol for behaviors
Ds	Change in displacement
Dv	Change in velocity
Dt	Change in time
DoD	Department of Defense
DPDU	DIS Detonation Protocol Data Unit
DSDE	Detect-Sort-Decide-Engage
E_{inner}	Inner Environment
E_{Outer}	Outer Environment

ESPDU	DIS Entity State Protocol Data Unit
FPDU	DIS Firing Protocol Data Unit
GIF	Graphics Interchange Format for image files
GIS	Geographic Information System
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HVU	High Value Unit (typical large surface naval vessel)
JPEG	Joint Photographic Entertainment Group format for image files
IP	Internet Protocol
ISO	International Standards Organization
IT	Information Technology
IT-21	Information Technology for the 21st Century
JNLP	Java Network Launch Protocol
LOE	Limited Objective Experiment
LSVE	Large Scale Virtual Environment
M&S	Modeling and Simulation
MAS	Multi-Agent System
NIMA	National Imagery and Mapping Agency
NATO	North Atlantic Treaty Organization
NPS	Naval Postgraduate School
OS	Operating System
PC	Personal Computer
PNG	Portable Network Graphics format for image files
RA	Reactive Agent

RHIB	Rigid Hull Inflatable Boat
SAVAGE	Scenario Authoring and Visualization for Advanced Graphical Environments
SCA	Symbolic Constructor Agent
SGML	Standard Generalized Markup Language
SVG	Scalable Vector Graphics
TM	Template Manager
UI	User Interface
USMTF	United States Message Text Format
USV	Unmanned Surface Vehicle
X3D	Extensible 3D Graphics
XHTML	Extensible Hypertext Markup Language
XMSF	Extensible Modeling and Simulation Framework
XML	Extensible Markup Language
XSLT	Extensible Stylesheet Language for Transformation
VRML	Virtual Reality Modeling Language
W3C	World Wide Web Consortium
WYSIWYG	What-you-see-is-what-you-get interface

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. MANUAL CONFIGURATION TASK LISTING

Manual Configuration Task Sheet

Participant Number:

Benchmark Task Number	
1	Starting the App (Make notes about user questions and ability to start the application) Once the app is started ask the participant to choose Manual Mode and then note the time. **START TIME**
2	Manually Choose harbor *(Choose Port Hueneme as the harbor for this scenario)* Good clicks/Key strokes Keyboard to mouse moves Errors
3	Manually Choose ship *(Choose DDG-51 Class ship for this scenario)* Good clicks/key strokes Keyboard to mouse moves Errors
4	Manually place the ship and small boat *(Tell the user which pier to place the ship along, and where to put the small boat.)* Good clicks/key strokes Keyboard to mouse moves Errors
5	Manually Choose a small boat attack *(Ask the user to select a terrorist small boat attack)* Good clicks/key strokes Keyboard to mouse moves

Errors

---

6 Manually place the terrorist at a start point of the attack path
(Ask the user to place the terrorist boat in a general location)

Good clicks/key strokes
Keyboard to mouse moves

Errors

---

7 Manually choose waypoints for the terrorist attack path
(Ask the user to place the terrorist and use 4 waypoints to designate the attack path)

Good clicks/key strokes
Keyboard to mouse moves

Errors

Once the last waypoint is placed, note the stop time.

STOP TIME

8 **Give the subject Q and A sheet #1**

APPENDIX C. MANUAL CONFIGURATION QUESTIONNAIRE

Q and A sheet 1

Participant Number:

Please mark your answer the following questions:

Overall, how easy was it to figure out how to configure the scenario you just completed?

1	2	3	4	5
(obvious)		(had to think and experiment)		(too hard, had to ask questions or use help screens)

How easy was it to select a Harbor and a Ship type?

1	2	3	4	5
(obvious)		(had to think and experiment)		(too hard, had to ask questions or use help screens)

How easy was it to place the Ship, small boat and terrorist boat?

1	2	3	4	5
(obvious)		(had to think and experiment)		(too hard, had to ask questions or use help screens)

If there were a step by step Wizard to guide the configuration, would you use it?

Yes	Maybe (It depends on the wizard)	No

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. WIZARD CONFIGURATION TASK LISTING

Wizard Configuration Task Sheet

Participant Number:

Benchmark Task Number		
	Restart the application (Make notes about user questions and ability to close and restart the application) Once the app is started ask the participant to choose Wizard and then note the time. **START TIME**	
9	Choose harbor *(Choose Port Hueneme as the harbor for this scenario)* Good clicks/keystrokes Keyboard to mouse moves Errors	
10	Choose ship *(Choose DDG-51 Class ship for this scenario)* Good clicks/keystrokes Keyboard to mouse moves Errors	
11	Place the ship and small boat *(Tell the user which pier to place the ship along, and where to put the small boat.)* Good clicks/keystrokes Keyboard to mouse moves Errors	
12	Choose a small boat attack *(Ask the user to select a terrorist small boat attack)* Good clicks/keystrokes Keyboard to mouse moves	

Errors

---

13 Place the terrorist at a start point of the attack path
(Ask the user to place the terrorist boat in a general location)

Good clicks/keystrokes

Keyboard to mouse moves

Errors

---

14 Choose waypoints for the terrorist attack path
(Ask the user to place the terrorist and use 4 waypoints to designate the attack path)

Good clicks/keystrokes

Keyboard to mouse moves

Errors

Once the last waypoint is placed, note the stop time.

STOP TIME

15 **Give the subject Q and A sheet #2**

APPENDIX E. WIZARD CONFIGURATION QUESTIONNAIRE

Q and A sheet 2

Participant Number:

Please mark your answer the following questions:

Overall, how easy was it to configure the scenario using the Wizard?

1	2	3	4	5
(obvious)		(had to think and experiment)		(too hard, had to ask questions or use help screens)

How easy was it to select a Harbor and a Ship type?

1	2	3	4	5
(obvious)		(had to think and experiment)		(too hard, had to ask questions or use help screens)

How easy was it to place the Ship, small boat and terrorist boat?

1	2	3	4	5
(obvious)		(had to think and experiment)		(too hard, had to ask questions or use help screens)

If you had to use this application again, would you use the Wizard?

Yes　　　Maybe　　　No

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX F. USER CHOICE CONFIGURATION TASK LIST

Users Choice Configuration Task Sheet

Participant Number:

Benchmark Task Number

Restart the application
(Make notes about user questions and ability to close and restart the application)

Once the app is started ask the participant to configure a scenario again and note the time.

START TIME

User chose Manual or Wizard?

16 Choose harbor
(Choose Port Hueneme as the harbor for this scenario)

Good clicks/keystrokes
Keyboard to mouse moves

Errors

17 Choose ship
(Choose DDG-51 Class ship for this scenario)

Good clicks/keystrokes
Keyboard to mouse moves

Errors

18 Place the ship and small boat
(Tell the user which pier to place the ship along, and where to put the small boat.)

Good clicks/keystrokes
Keyboard to mouse moves

Errors

19 Choose a small boat attack
(Ask the user to select a terrorist small boat attack)

Good clicks/keystrokes

Keyboard to mouse moves

Errors

20 Place the terrorist at a start point of the attack path
(Ask the user to place the terrorist boat in a general location)

Good clicks/keystrokes

Keyboard to mouse moves

Errors

21 Choose waypoints for the terrorist attack path
(Ask the user to place the terrorist and use 4 waypoints to designate the attack path)

Good clicks/keystrokes

Keyboard to mouse moves

Errors

Once the last waypoint is placed, note the stop time.

STOP TIME

22 **Give the subject Q and A sheet #3**

APPENDIX G. USER CHOICE CONFIGURATION QUESTIONNAIRE

Q and A sheet 3

Participant Number:

Please mark your answer the following questions:

Overall, how easy was it to figure out how to configure the scenario you just completed?

1 (obvious)	2	3 (had to think and experiment)	4	5 (too hard, had to ask questions or use help screens)

How easy was it to select a Harbor and a Ship type?

1 (obvious)	2	3 (had to think and experiment)	4	5 (too hard, had to ask questions or use help screens)

How easy was it to place the Ship, small boat and terrorist boat?

1 (obvious)	2	3 (had to think and experiment)	4	5 (too hard, had to ask questions or use help screens)

Did you use the Wizard to guide this last configuration?

Yes	No, didn't want to use the wizard.	No, didn't know I could use the wizard.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX H. MISCELLANEOUS FUNCTION TASK LISTING

MISC Functions Task Sheet

Participant Number:

Benchmark Task Number		
	Once the participant has the misc task list, note the time.	
	START TIME	
23	Print the scenario just constructed	
	Good clicks/keystrokes	
	Keyboard to mouse moves	
	Errors	
24	Save the scenario just constructed	
	Good clicks/keystrokes	
	Keyboard to mouse moves	
	Errors	
26	Run and Record the scenario	
	Good clicks/keystrokes	
	Keyboard to mouse moves	
	Errors	
27	Change 3D viewpoint to Rhib	
	Good clicks/keystrokes	
	Keyboard to mouse moves	
	Errors	
28	Change 3D viewpoint to Terrorist Boat	
	Good clicks/keystrokes	
	Keyboard to mouse moves	
	Errors	

29	Change 3D viewpoint to DDG Top View		
	Good clicks/keystrokes		
	Keyboard to mouse moves		
	Errors		
	START TIME		
29			**Give the subject Q and A sheet #4**

APPENDIX I. MISCELLANEOUS FUNCTION TASK QUESTIONNAIRE

Q and A sheet 4

Participant Number:

Please mark your answer the following questions:

How important is it to be able to print a 2D view of an AT/FP Scenario?

1	2	3	4	5
(vital)		(may be useful, but could live without it)		(I would never print out the 2D view)

How important is it to be able to save and re-use AT/FP Scenarios?

1	2	3	4	5
(vital)		(may be useful, but could live without it)		(I would never save the scenario)

How easy was it to run the AT/FP Scenario?

1	2	3	4	5
(obvious)		(had to think and experiment)		(too hard, had to ask questions or use help screens)

How important is it to be able to save and replay 3D simulation runs?

1	2	3	4	5

(vital)		(may be useful, but could live without it)		(I would never save a 3D simulation run for future viewing)

How easy was it to figure out how to save a 3D simulation run?

1	2	3	4	5
(obvious)		(had to think and experiment)		(too hard, had to ask questions or use help screens)

APPENDIX J. JAVA PROGRAMMING UTILITIES

A. INTRODUCTION

This section provides an overview of pertinent Java programming utilities either reused or developed in the context of this thesis that may prove useful to one interested in development details.

B. SCREEN CAPTURING AND BASIC IMAGE MANIPULATION IN J2SDK1.4.1_X

In the context of this thesis, we faced the problem of how to save the state of scenario configuration in 2D post-scenario HTML presentation of results to the end-user. The resulting solution was to investigate current methods utilized to affect a basic screen capture to a standard image format, such as jpeg. This subsection reviews the process undertaken to accomplish this task.

```
import java.awt.*;
import java.awt.image.BufferedImage;
import java.awt.geom.*;
import java.awt.geom.Point2D;
import java.io.File;
import java.lang.reflect.Array;
import java.net.URL;
import java.util.*;
import javax.imageio.ImageIO;
import javax.swing.*;
import com.sun.image.codec.jpeg.*;
```

The first item of note is that imaging capabilities introduced in the j2sdk1.4.0 and further refined in j2sdk1.4.1_0x from Sun Microsystems were utilized. Also, the com.sun.image.code.jpeg package is utilized for writing out to the jpeg image format. Although this package has been exposed to developers for usage for several years, there is a standard disclaimer provided by Sun that they retain the right to change, modify, or remove any or all com.sun package implementations, so that when changing to a new version of Java one should check the API documentation for this package to see what has changed.

```
public class ImageCreator
    extends Object {
```

The first few items to be examined are the data members of the class that we are reviewing. First, we maintain an instance of a BufferedImage object. The BufferedImage subclass of the java.awt.Image object, describes an Image that has an accessible buffer of image data. It is made of a ColorModel and Raster of image data. [SUN 2003] Next, we maintain a Graphics2D object for painting the image we want to create in an off-screen fashion. The Graphics2D class provides an extension of the Java Graphics class in order to provide finer grained control to the developer over the geometry, transformation of coordinates, text layout, and color management. It is considered to be the fundamental class for rendering 2D on the Java platform. [SUN 2003]

```
        private BufferedImage _outputImage;
        private Graphics2D    _g2dOutput;
```

Next, we have a few attribute values that will be necessary for writing out a jpeg encoded image. First is the image quality that we want the output image to be. This value ranges from zero to one inclusive, with numbers closer to one being of higher quality. Then, the x and y dimensions in pixels that we want the output image to be. And lastly, the String based output filename to utilize for the location and file name of the output image created.

```
        private float  _imageQuality;
        private int    _xDimension;
        private int    _yDimension;
        private String _outputFileName;
```

Next, we declare storage vectors for the type of components we want to be able to paint to our image being created in addition to a corresponding location vector that holds the absolute 2D X-Y position in which to render the desired component. The 2D position is treated as the upper left corner of the component's location, with the top left corner of the produced image being considered to be the coordinates (0,0). In this implementation, we treat the components storage vectors as acting in a first-in-first-out manner, but any sensible variation of implementation should suffice. Additionally, in this implementation, we allow for the idea that the user might want to paint individual points

on the output image for items such as track reconstruction in 2D in a simulation, so an additional storage vector for points and corresponding colors to paint is incorporated.

```
private Vector _componentVector;
private Vector _componentLocation;
private Vector _pointVector;
private Vector _pointColorVector;
```

Next, we have the default no argument constructor for the class which has one job; to provide legal default initialization values for the components of the class in the case of one or more not being set by the developer utilizing the class and initializing the storage vectors.

```
public ImageCreator() {
    this.setImageQuality( .8f );
    this.setOutputImageSize( 700, 460 );
    this.setOutputFileName( "test.jpg" );
    init();
}

private void init() {
    _componentVector = new Vector();
    _componentLocation = new Vector();
    _pointVector = new Vector();
    _pointColorVector = new Vector();
    _outputImage = new BufferedImage( _xDimension,
             _yDimension, BufferedImage.TYPE_INT_RGB );
    _g2dOutput = _outputImage.createGraphics();
}
```

We then include basic access methods for setting the private data members defined earlier such as the image quality, image size, output filename, and so on. Of note, for this implementation we take basic care to ensure one can't set an image property to a an illegal state such as a negative size, quality setting outside of the legally defined range, etc. Additionally, we define the methods for adding components and points to paint, as well as methods for clearing previously added components in the event of reuse of a single object created for outputting images.

```
public void setImageQuality( float pQuality ) {
    if ( pQuality < 0.0f ) {
        _imageQuality = 0f;}
    else if ( pQuality > 1.0f ) {
        _imageQuality = 1.0f; }
    else{
        _imageQuality = pQuality;} }

public void setOutputFileName( String pOutputFileName ) {
```

```
            _outputFileName = pOutputFileName;}

        public void setOutputImageSize( int pXdimension, int pYdimension ) {
            if ( pXdimension < 0 ){
                _xDimension = 0;}
            else{
                _xDimension = pXdimension; }
            if ( pYdimension < 0 ) {
                _yDimension = 0; }
            else {
                _yDimension = pYdimension; } }


        public void addComponentToPaint( URL pURL ) {
            _componentVector.add( pURL );}

        public void setComponentLocation( AffineTransform pTransform ) {
            _componentLocation.add( pTransform );}


        public void addPointToPaint( Point2D pPoint, Color pColor ) {
            _pointVector.add( pPoint );
            _pointColorVector.add( pColor );}

        public void clearComponentsToPaint() {
            Vector emptyVector = new Vector();
            _componentVector = emptyVector;
            Vector emptyVector2 = new Vector();
            _componentLocation = emptyVector2;
        }

        public void clearPointsToPaint() {
            Vector emptyVector1 = new Vector();
            Vector emptyVector2 = new Vector();
            _pointVector = emptyVector1;
            _pointColorVector = emptyVector2;

        }



        public void resetAllComponets() {
            this.setImageQuality( .8f );
            this.setOutputImageSize( 800, 600 );
            this.setOutputFileName( "test.jpg" );
            this.init();
        }
```

Next, we define our writeImage method, which is responsible for writing the information currently contained or passed to our class out as a JPEG encoded image.

```
        public void writeImage() {
```

```
            try
            {
```

First, the method invokes the utility method createImage which paints the components and points contained in the storage vectors previously defined in the class.

```
                this.createImage();
```

The createImage utility method first paints the componentVector in a first-in-first-out manner to the Graphics 2D Object that is being used to paint to our BufferedImage for making the JPEG file, then paints any points that have been passed following the components so that they are visible on the output image. Order matters when painting to the Graphics2D Object. The items can be thought of as being layered on top of one another in the order painted.

```
        private void createImage() {

            for ( int idx = 0; idx < _componentVector.size(); ++idx
             ){
                try
                {
                 URL tempURL = ( URL ) _componentVector.get( idx );
                  BufferedImage myImage = ImageIO.read( tempURL );
                    AffineTransform tempTransform = (
                     AffineTransform ) _componentLocation.get( idx );
              _g2dOutput.drawImage( myImage, tempTransform, null );}
            }
            for ( int idy = 0; idy < _pointVector.size(); ++idy )
            {
               Point2D.Double tempPoint = ( Point2D.Double )
                     _pointVector.get( idy );
                _g2dOutput.setColor( ( Color )
                     _pointColorVector.get( idy ) );
                _g2dOutput.draw( new Rectangle( ( int )
                     tempPoint.getX(), ( int )
                     tempPoint.getY(), 1, 1 ) ); }       }
```

Then, we open a FileOutputStream object with the desired or default output file name for creating the JPEG followed by creating a JPEGImageEncoder instance with this FileOutputStream instance.

```
                FileOutputStream out = new FileOutputStream(
                     _outputFileName );
                JPEGImageEncoder encoder =
                     JPEGCodec.createJPEGEncoder( out );
```

Next, we use our BufferedImage that has been painted to by all of our desired components to create a JPEGEncodeParam object, and follow this by setting the image quality, and ultimately encoding the image, followed by closing the outputstream we have previously opened.

```
                JPEGEncodeParam param =
                 encoder.getDefaultJPEGEncodeParam( _outputImage );
                param.setQuality( _imageQuality, false );
                encoder.setJPEGEncodeParam( param );
                encoder.encode( _outputImage );
                out.close();  }
            catch ( Exception e ){}
        }

```

Finally, we show an exemplar implementation that draws an overhead image of Port Hueneme followed by a simple 2D image of a destroyer and then several blue points drawn down the left side of the image (Figure 96).

```
        public static void main( String args[] ) {
            ImageCreator myCreator = new ImageCreator();
            myCreator.setOutputFileName( "c:/test01.jpg" );
            URL chartURL = null;
            URL shipURL = null;
            try
            {
                chartURL = new URL(
                     "file:/c:/atfp/images/PortHuenemeChart.gif" );
            shipURL = new URL(
               "jar:file:/c:/atfp/images/images.jar!/ddgIcon.GIF" );
            }
            catch ( Exception e ){}

            myCreator.addComponentToPaint( chartURL );
            myCreator.setComponentLocation( new AffineTransform() );
            myCreator.addComponentToPaint( shipURL );
            AffineTransform tempTransform = new AffineTransform();
            tempTransform.setToTranslation(300,300);
            myCreator.setComponentLocation( tempTransform );
            myCreator.addPointToPaint( new Point2D.Double( 100, 100
                     ), Color.blue );
            myCreator.addPointToPaint( new Point2D.Double( 100, 101
                     ), Color.blue );
            myCreator.addPointToPaint( new Point2D.Double( 100, 102
                     ), Color.blue );
            myCreator.addPointToPaint( new Point2D.Double( 100, 103
                     ), Color.blue );
            myCreator.addPointToPaint( new Point2D.Double( 100, 104
                     ), Color.blue );
            myCreator.addPointToPaint( new Point2D.Double( 100, 105
                     ), Color.blue );
```

```
            myCreator.addPointToPaint( new Point2D.Double( 100, 106 ), Color.blue );
            myCreator.addPointToPaint( new Point2D.Double( 100, 107 ), Color.blue );
            myCreator.addPointToPaint( new Point2D.Double( 100, 108 ), Color.blue );
            myCreator.addPointToPaint( new Point2D.Double( 100, 109 ), Color.blue );
            myCreator.writeImage();
        }
```

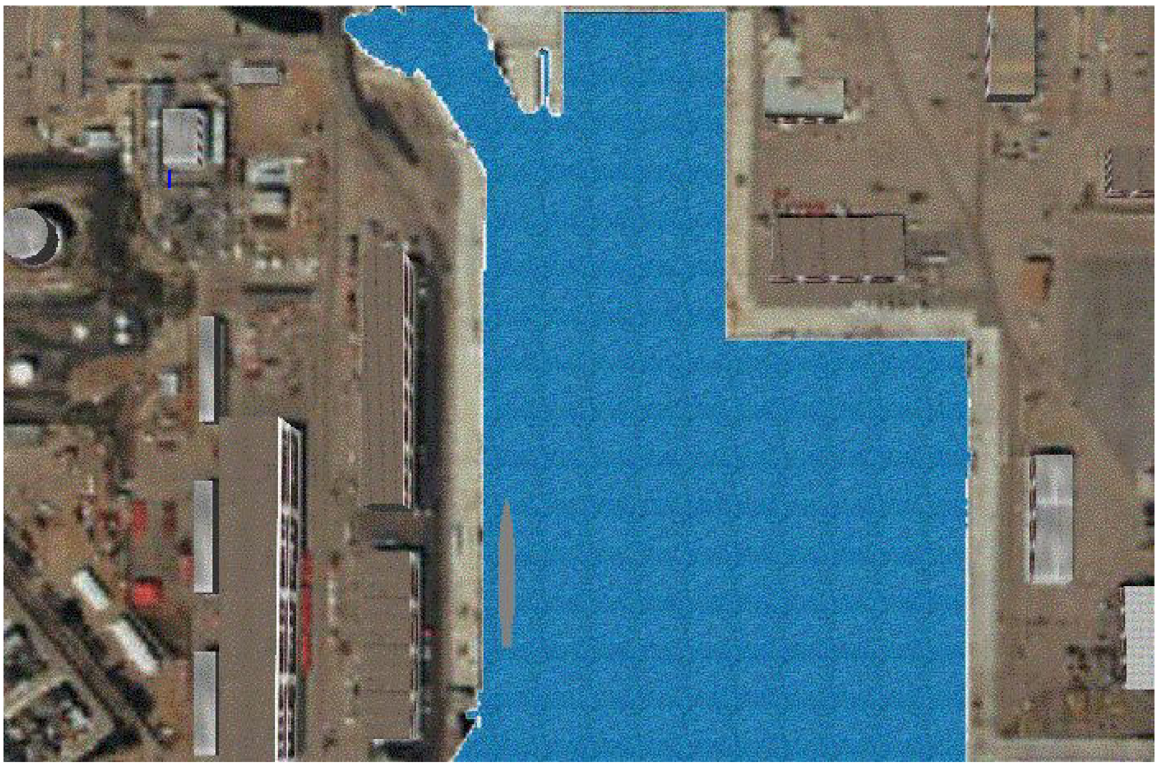


Figure 96. Depicts example output of the ImageCreator JPEG writer class.

So, the next extension not discussed in this appendix, is to add functionality for painting Java Components to the mix. Modifications to do so include painting the Java components first in the writeImage method. With the modifications made, results such as those depicted in Figure 97 can be achieved.

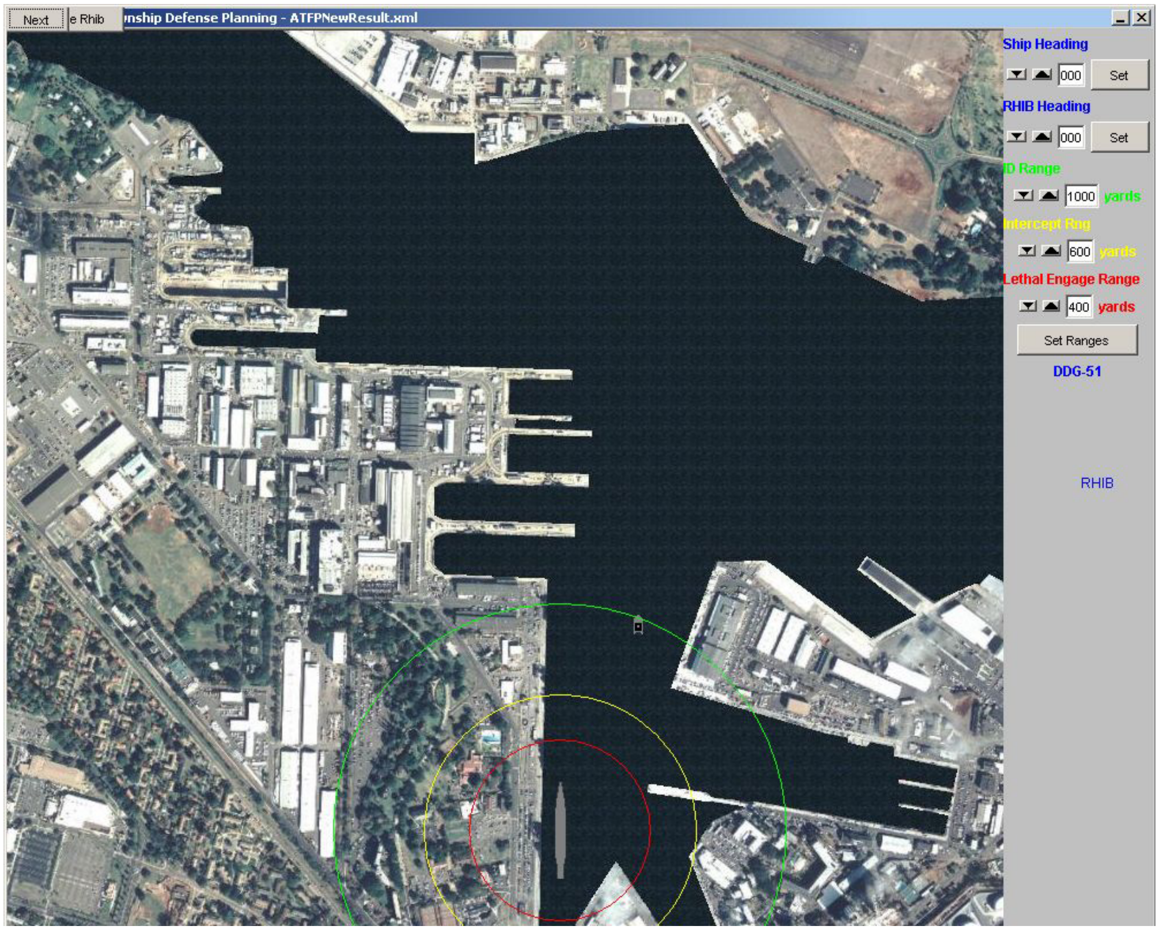


Figure 97. Depicts the resulting jpeg encoded output file obtained after adding Java Swing Components.

C. PRINTING IN JAVA

Once one has done Image manipulation in Java, printing mainly involves shifting one's paradigm of painting objects to a file, to that of painting them to a Graphics object to send to the operating systems print stream. The primary interface of concern is the java.awt.print interface. This interface defines the print method, which takes as arguments a Graphics object instance, PageFormat object instance, and an int representing the pageIndex to be utilized and throws a PrinterException if the Print object aborts the printing job for any reason. [SUN 2003] The PageFormat class is used to describe the orientation and size of the page to be printed. [SUN 2003]

The next class we are concerned with when trying to print from our Java application is the PrintJob class. This class is invoked for setting up a print job, as well

as to invoke a printing dialog with the end user, then to print the pages of the job. Depicted in Figure 98, the Print and Print Setup buttons are exposed to the end user for selection on applicable screens while configuring an AT/FP scenario. In this case when viewing platform specific information on the DDG-51 Arleigh Burke class destroyer.

Figure 98. Depicts the Print and Print Setup buttons in the AT/FP Scenario Generator application in the upper right portion of the Ship Information panel.

Next, the page setup dialog (Figure 99) and the print setup dialog (Figure 100) are depicted.

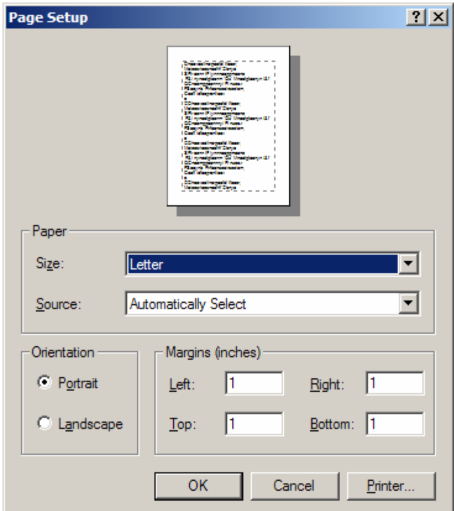


Figure 99. Depicts the Page Setup dialog.

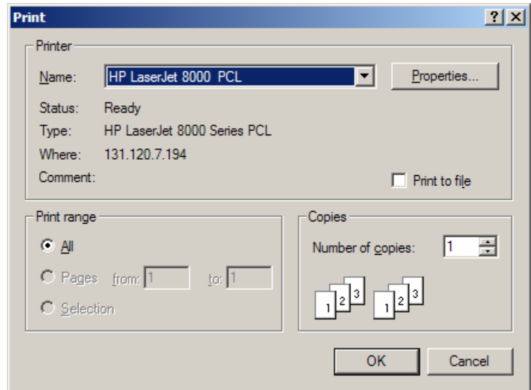


Figure 100. Depicts the Print Setup dialog.

One problem encountered upon initial usage of the print interface was that the Graphics object did not allow for the concept of scaling. Upon further research, tutorials were found at http://www.javapro.com (accessed January 2003) and http://www.javadevelopers.com (accessed January 2003) that reviewed a basic process for leveraging the JEditorPane and Graphics2D classes for scaling the desired document for printing to a standard 8.5 inch by 11 inch format. This process basically consists of first casting the Graphics object instance in the print method to a Graphics2D object. Then using a JEditorPane, set it to the width of the printable page, and if it cannot due to the page being too large, scale the page to fit. This is done by clipping the bounds of the Graphics2D object to the size of the printable page we are looking at. For additional implementation details, the reader is referred to the online tutorials or to the source code for this thesis.

D. SUMMARY

In summary, the reader has been exposed to basic imaging operations in context of creating basic screen captures and saving as JPEG encoded files as well as a basic introduction to printing in the Java API.

APPENDIX K. APPLYING XSLT TECHNOLOGIES IN THE JAVA PROGRAMMING LANGUAGE

A. INTRODUCTION

This appendix reviews the basic syntactical and other requirements necessary in order to apply XML stylesheets to XML documents to produce HTML, text, and other XML instance files.

B. UTILIZING XSLT IN CLIENT-SIDE JAVA

When conducting a general literature review of the usage of XML and XSLT with the Java programming language, one can find a lot of material on its usage for Server-side programming. When conducting client-side programming it can generally be confusing for aspiring developers on how to utilize the many available XML libraries to effect basic transformations within the context of their Java applications. For this reason, inclusion of an example of the basic usage of XSLT with the java.xml.parser, java.xml.transform, org.w3d.dom, and org.xml.sax java libraries is reviewed. Additionally, the reader is referred to [Kay 2001], [MANGANO 2001], and [BURKE 2003] for further information.

C. JAVA AND XSLT EXAMPLE

The below example is a subset of the HTMLMaker Java class included in the source distribution of this thesis based on work provided by the draft work in [NEUSHAL 2003] geared towards GIS solutions.

First, we list the library imports necessary for this example. Of note, there are several different ways to effect transformation with XSLT in Java, and it is recommended to the reader to review current best practices available at http://www.sun.com/developers (accessed February 2003), http://www.jdom.org (accessed February 2003), as well as investigation into any emerging XML libraries or technologies not discussed in this thesis.

```
import javax.xml.parsers.DocumentBuilder;
```

```
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.FactoryConfigurationError;
import javax.xml.parsers.ParserConfigurationException;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
import org.w3c.dom.Document;
import org.w3c.dom.DOMException;

// For write operation
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamSource;

import javax.xml.transform.stream.StreamResult;

import java.io.*;

```

First, we need to look at the over-all purpose of this class. We wrote this class to do one thing, given a string based URL for an XSLT, input filename, and output filename, effect an XSLT transform on the input file. So, we see that we declare our class to have one data member, a static instance of a DOM Document object. This is used to store the input data file once we parse it to XML.

```
public class HTMLMaker {

    static Document document;


    public HTMLMaker( String xsl, String filename, String
         newfilename ) {
```

In this example, we are doing all of the work in the constructor of the class. If reusing for production purposes, typically get and set methods are exposed so that one doesn't have to create new object instances continually. In the context of our use in this thesis, the class has only one instance generated per scenario run and did not prove to be of consequence that we created multiple ones. So, the first thing we do in the constructor is to create an instance of a Document BuilderFactory from the java.xml.parsers library. The DocumentBuilderFactory provided from the Sun libraries provides a factory API that lets one obtain a parser to produce DOM trees from XML documents. [SUN 2003] This class is not guaranteed to be thread safe, so when deploying in a multi-threaded environment, we would need to ensure this on our own. In this context though, we are

not accessing the XML input file in another thread so there is no need to worry about this case. After creating a new factory object instance, we want to make sure that it is namespace aware in the case of using multiple namespaces within an XML instance document (say our native schema combined with the X3D schema and Generic Hub).

```
            DocumentBuilderFactory factory =
            DocumentBuilderFactory.newInstance();
            factory.setNamespaceAware( true );
```

Next, we create file objects for our XSLT, input XML file, and the desired output file from the java.io library.

```
            try
            {
                File stylesheet = new File( xsl );
                File datafile = new File( filename );
                File newfile = new File( newfilename );
```

Our next step is to create a DocumentBuilder object in which to parse our input XML file into the static document instance mentioned previously. Once we obtain an instance of this class, we can parse our input from other sources besides the File instance shown here. We could also use InputStreams, URL's, or SAX InputSources if desired. [SUN 2003]

```
                DocumentBuilder builder =
                     factory.newDocumentBuilder();
                document = builder.parse( datafile );
```

The next step is to create and get a handle to an instance of a TransformerFactory object instance in order to apply our XSLT to the XML file we now have in memory as a DOM object instance. The TransformerFactor is part of the java.xml.transform package and is used to generate Transformer and Template class object instances. [SUN 2003]

```
                // Use a Transformer for output
                TransformerFactory tFactory =
                     TransformerFactory.newInstance();
```

Next, we create a new StreamSource instance from the XSLT File object that we have. The StreamSource object does one thing for us; it acts as a placeholder for our transformation source, but in the form of a stream of XML. [SUN 2003]

```
            StreamSource stylesource = new StreamSource(
                stylesheet );
```

Then, we use the StreamSource information in which to seed our Transformer instance object created from the previous factory implementation.

```
            Transformer transformer = tFactory.newTransformer(
                stylesource );
```

Then, we use the Document instance that we created earlier to create a new DOMSource instance object. The DOMSource class simply acts as a holder for the transformation source tree in the form of a DOM tree that we will use again shortly. [SUN 2003] The final item that we create is a StreamResult instance from the output File instance that we've created from the desired parameters passed to the constructor of the class. An object which implements this interface simply is designed to contain the information which is required by our underlying XML libraries to create a resulting transformation tree from the application of an XSLT.

```
            DOMSource source = new DOMSource( document );
            StreamResult result = new StreamResult( newfile );
```

Finally, we pass the source and result instance objects to our transformer instance object in which to process the source tree to the result tree for us.

```
            transformer.transform( source, result );

        }
```

Now, we list the various catch statements that although not required, can be considered good practice to include when we might encounter problems with trying to apply XSLT to instance documents.

```
        catch ( TransformerConfigurationException tce )
        {}
        catch ( TransformerException te )
        {}
        catch ( SAXException sxe )
```

```
            {}
            catch ( ParserConfigurationException pce )
            {}
            catch ( IOException ioe )
            {}
```

The final portion of the example shown is a simple test case of applying an XSLT to an instance document with a desired filename on a Windows Operating System. Improvements to this class can be made in the exposure of all of the available input sources as separate class arguments for the XSLT, input XML file, and output file in addition to examining requirements for usage in a thread safe manner.

```
        public static void main( String args[] ) {
            HTMLMaker myMaker = new HTMLMaker(
                     "c:/atfp/xml/xsl/ScenarioToHTML.xsl",
                     "c:/atfp/xml/output/ATFPNewResult.xml",
                     "c:/atfp/xml/htmloutput/atfp01.html" );
        }
```

E. SUMMARY

In summary, the reader has been exposed to the basic utilization of the core Java XML library APIs in order to apply an XSLT to an XML instance document within a client-side Java application. Acknowledgements go to Captain James Neushul, USMC, who provided initial instruction and source to the author for this area of development.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX L. JAVA APPLICATION INSTALLER CREATION WITH ZEROG.COM INSTALL ANYWHERE

A. INTRODUCTION

This appendix provides an overview of how to create installation programs for Java applications with the commercial application, Install Anywhere by ZeroG.com of San Francisco, California. The focus of the appendix will be on the advanced designer option provided by the application with exploration of the basic designer left to the reader.

B. MOTIVATION FOR USE

As outlined previously in this thesis, there was substantial motivation to investigate current industry best practices for the deployment of Java technologies for use on the client without being able to guarantee nor deploy a current Java Virtual Machine to end-user's web browsers. The Install Anywhere Enterprise Edition Java deployment solution from ZeroG was found to be in use by industry (http://www.borland.com) (accessed March 2003), and based on open architectures and utilizing open standards [ZEROG 2002], whilst deploying a commercial product.

C. CREATING THE INSTALLATION APPLICATION

The first step when creating a new application installer is to either select a new project or an existing one (Figure 101). The Install Anywhere project files are simply XML documents that contain the various properties used for configuration of a previous or the currently being configured installer. Once this has been done, the next item to configure is the general project information.

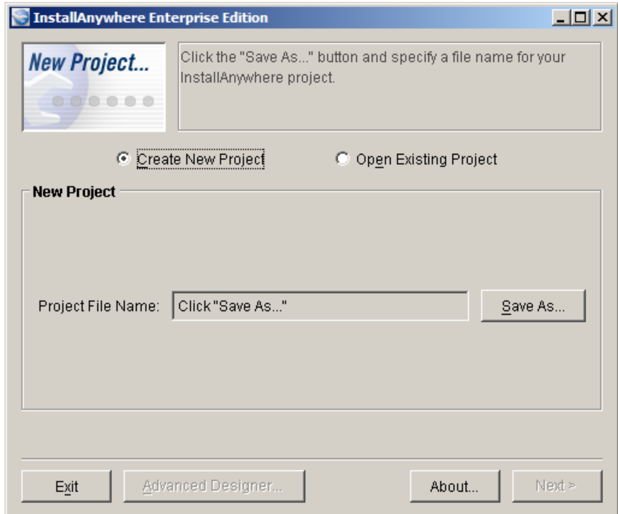


Figure 101. Depicts the startup screen for the Install Anywhere Enterprise Edition application.

General project information is comprised of the project title, location to store the XML-based project configuration file, as well as the project title to utilize on the deployment applet that will be generated when the installation creation is complete (Figure 102).

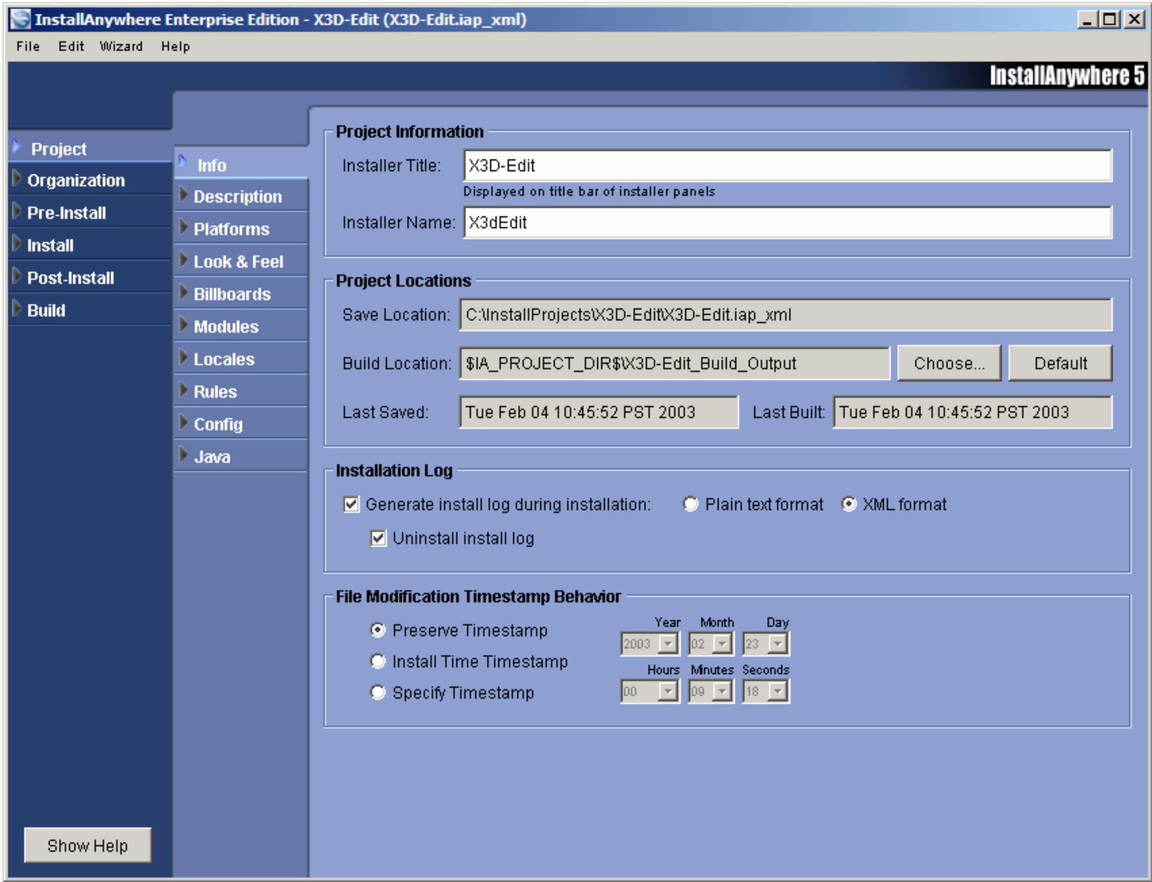


Figure 102. Depicts the Installer Information Configuration panel for Install Anywhere.

The next step is to configure the project description panel (Figure 103). Items included in this panel are things such as the project description, home web page, versioning and time stamp information, project information web site, as well as the point of contact for the application with email address if desired.

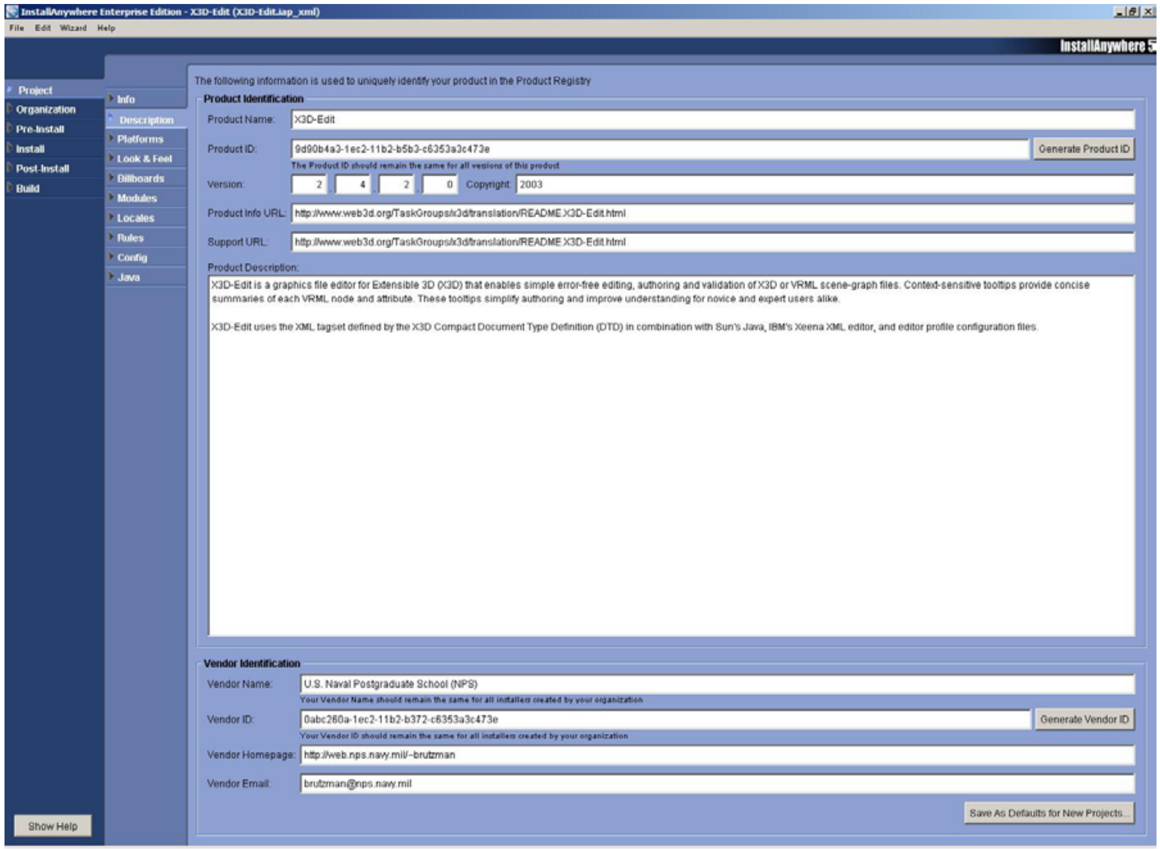


Figure 103. Depicts the Project Description Configuration panel for Install Anywhere.

Next, configuration information is selected for the various platforms that are being deployed to for the current installer creation (Figure 104).

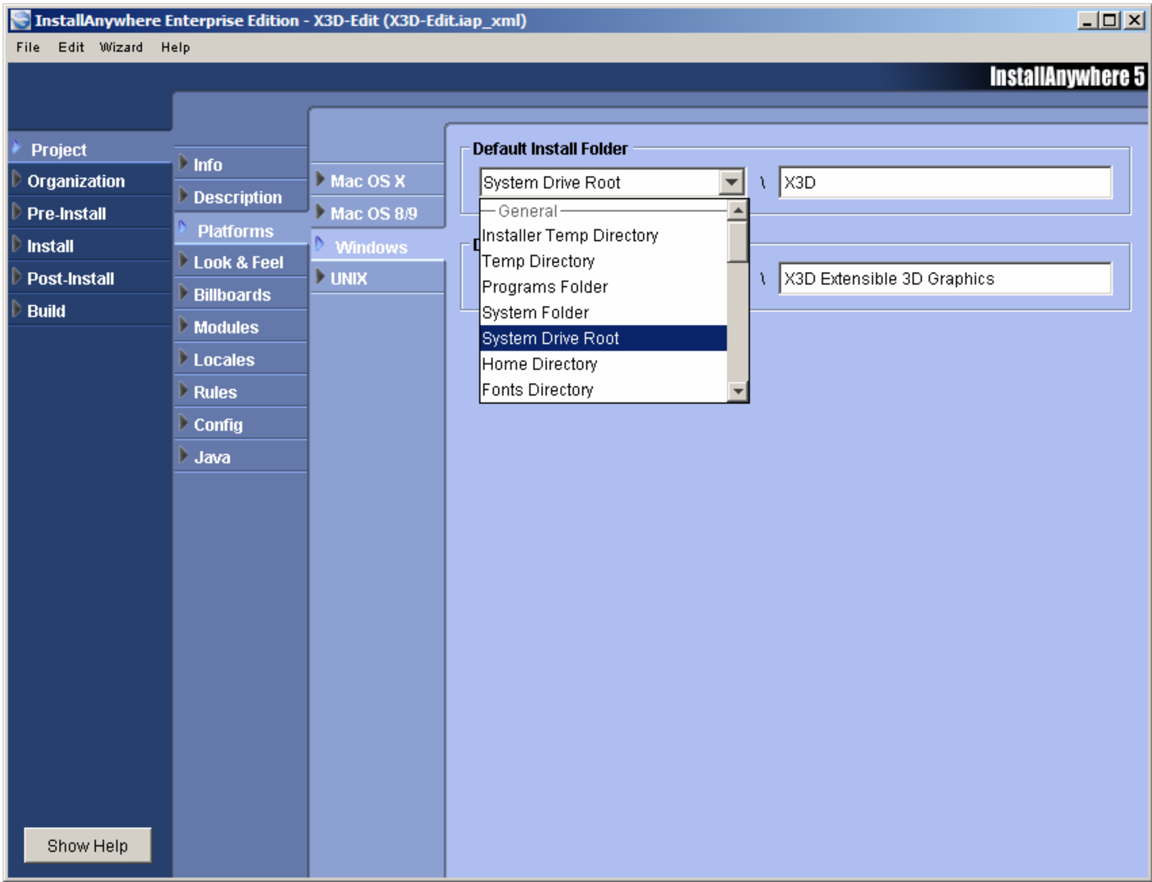


Figure 104. Depicts the Platform Configuration panel for Install Anywhere.

The next few options in the advanced installer are concerned with the configuration of the look and feel of the display panels that will be presented to the end-user when running the installation program. Figures 105 and 106 depict the configuration of the startup splash screen image and installation panel billboards for the X3D-Edit installation program configuration. It was found that for both the X3D-Edit and AT/FP Scenario Generator application installers it was necessary to go through a series of trial and error steps in order to find the best image size and quality to utilize for final installer creation. There was a capability to preview the images selected when creating the installation programs, but this did not always provide an accurate feeling for what the actual result would be when generating the final installation application.

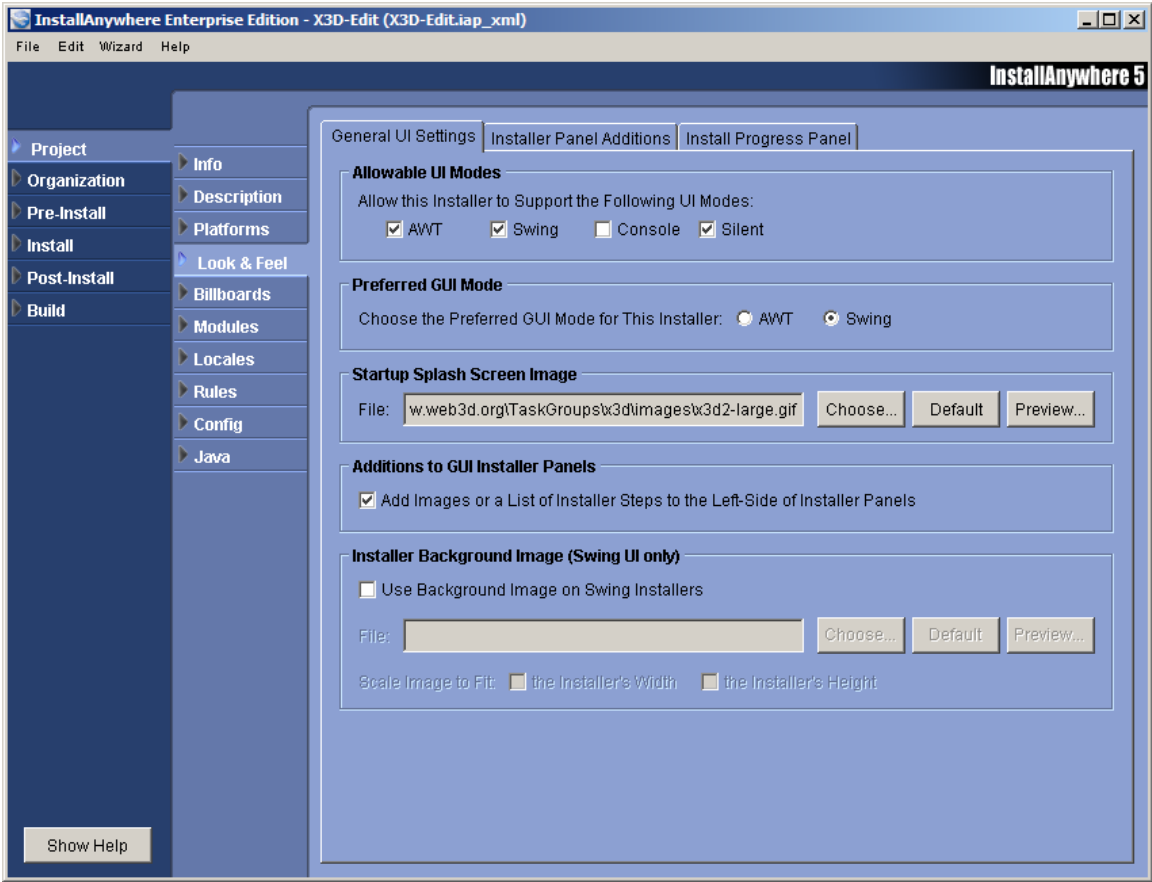


Figure 105. Depicts the Look and Feel Configuration panel in Install Anywhere.

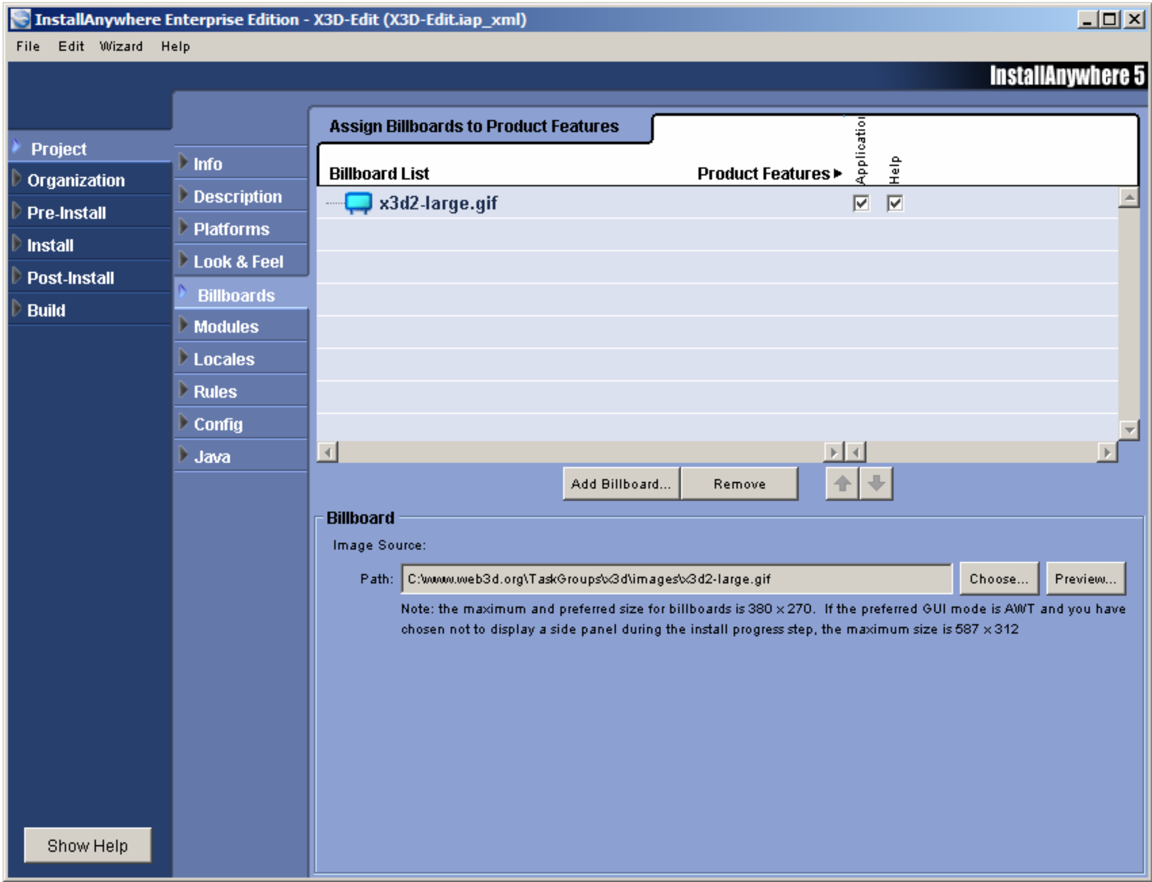


Figure 106. Depicts the Billboards configuration for Install Anywhere.

The next item to complete while generating the installation application was to select the support locales with respect to language support in the deployed application (Figure 107). The X3D-Edit scene graph editing application supports multi-lingual tool-tips in the French, German, and Spanish languages with ongoing work in the Iranian and Portuguese languages. As a result, as the tool-tip work is completed for a language, the applicable locale is added to the installation application configuration in order to better support end-users that do not speak English natively. For the three languages with tool-tip sets completed, we found the installer translation for steps to take to be adequate. When adding additional panels though, we found that one must incorporate multi-lingual capabilities in the HTML pages generated for inclusion since the application will only conduct this translation for the languages chosen on the default panel instructions.

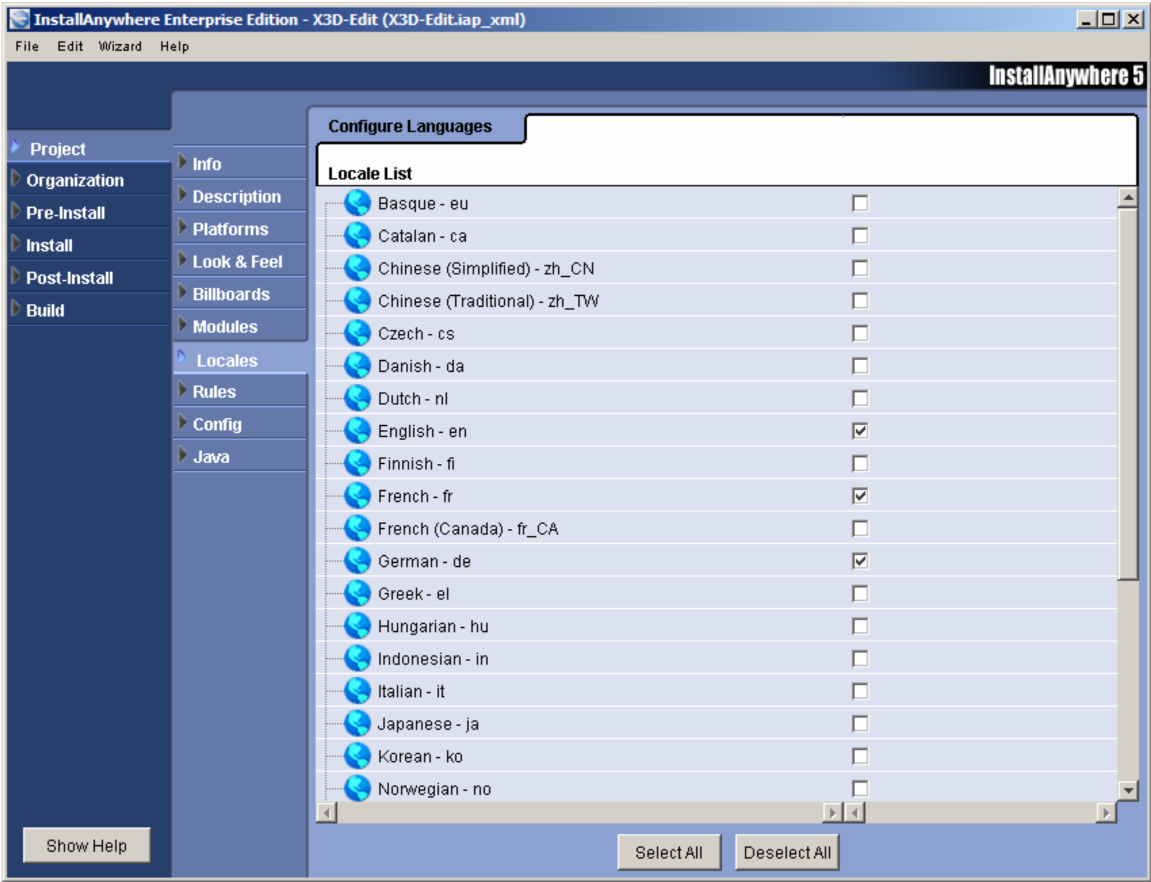


Figure 107. Depicts the Locale Configuration panel for Install Anywhere.

The next item to configure is the general configuration panel for the application installer being developed (Figure 108). The first item to note is that by selecting to send the standard output to the console, we can keep the look and feel of a Java application being run which is often desired for various Java applications that are being developed in conjunction with emerging technologies. When looking to deploy to a larger client base in a production-type style application, this may not be generally desired since we do not want to confuse the end-user more than is necessary. Additionally, the minimum and maximum Java heap sizes may be configured here if one does not desire to utilize the 64MB default provided by a standard Java Virtual Machine (at the time of this writing). The last option on this panel to mention is the allowable VM listing. Even though we may configure this here, it is generally easier to select/configure this on a later panel.

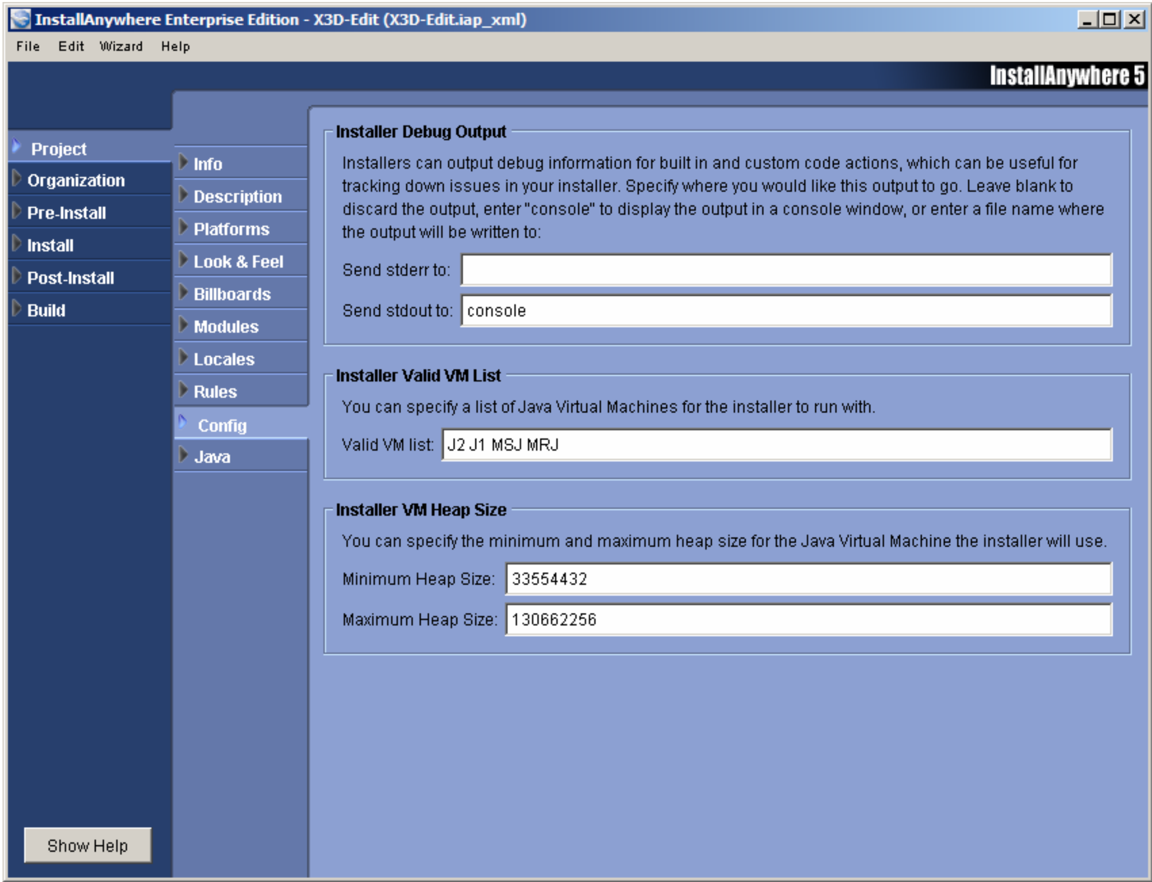


Figure 108. Depicts the General Application Configuration panel for InstallAnywhere.

The next step we are concerned with is the configuration of the Pre-Installation Action list (Figure 109). For the pre-installation actions and panels that the user will see, we can add additional panels, change the wording of existing ones, and so forth.

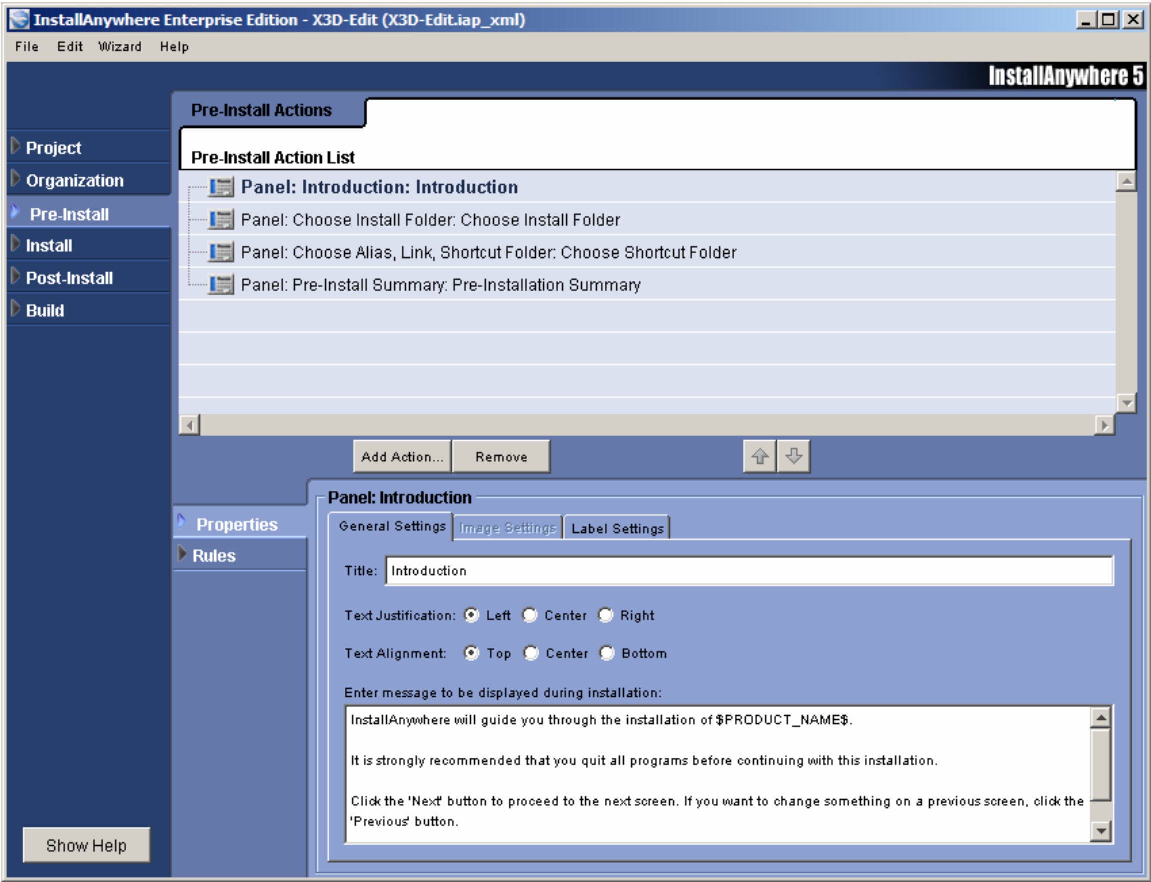


Figure 109. Depicts the Pre-Installation Action configuration panel in InstallAnywhere.

The next step is probably the most important. It is the installation configuration panel (Figure 110). In this panel, we select the location of deployment for the application and support files. Also, we configure the shortcuts and application launchers that will be exposed for the end-user to run after application delivery. Additionally, we can add any registry entries for Windows operating system deployments that might be necessary for the targeted client machines (in this case a simple registry entry that associates the X3D icon with all X3D files on the client machine).

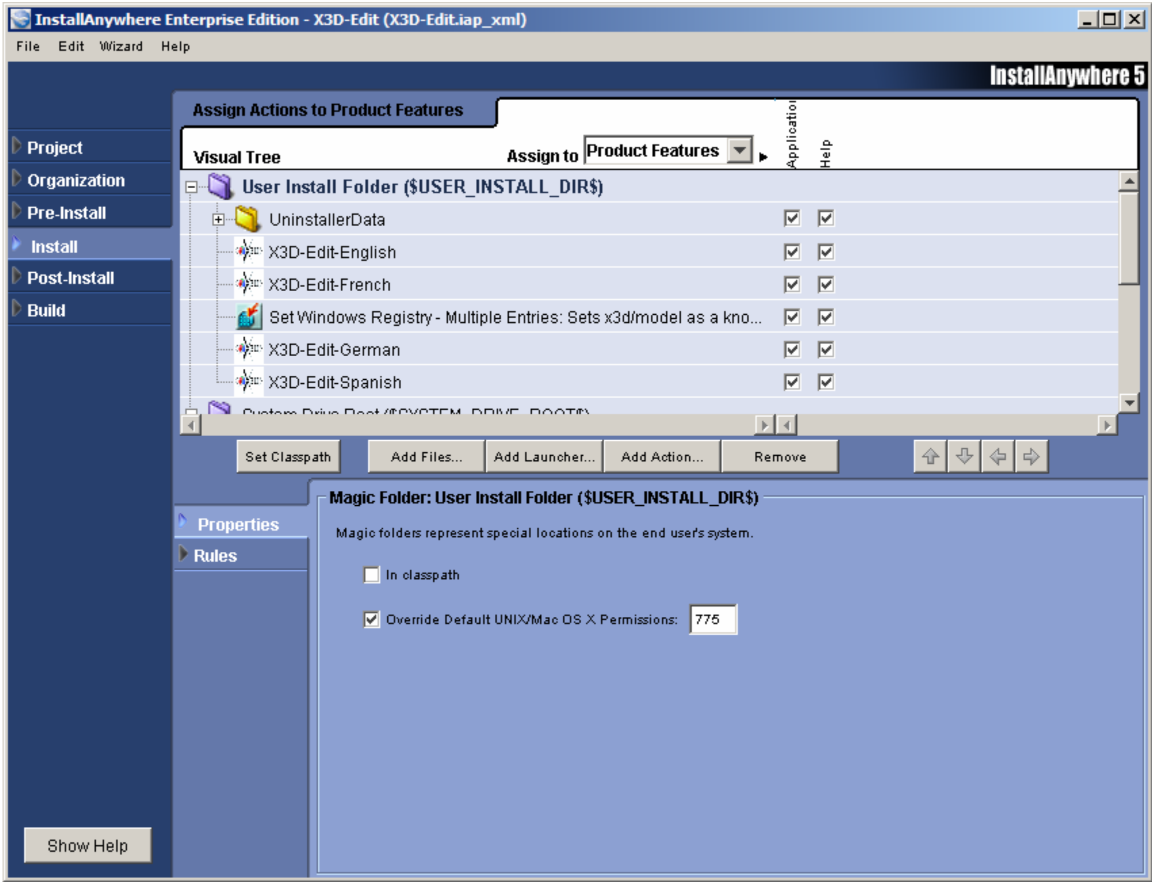


Figure 110. Depicts the Installation Configuration panel for InstallAnywhere.

Figure 111 depicts how we configure Java main class arguments for our deployed application to be utilized in an Operating System independent manner for X3D-Edit. We make use of the exposure of system variables to us through the Install Anywhere environment in order to reference the System Drive Root, vice the use of a lettered drive which would then make it necessary for us to repeat installation creation work for each operating system desired for deployment.

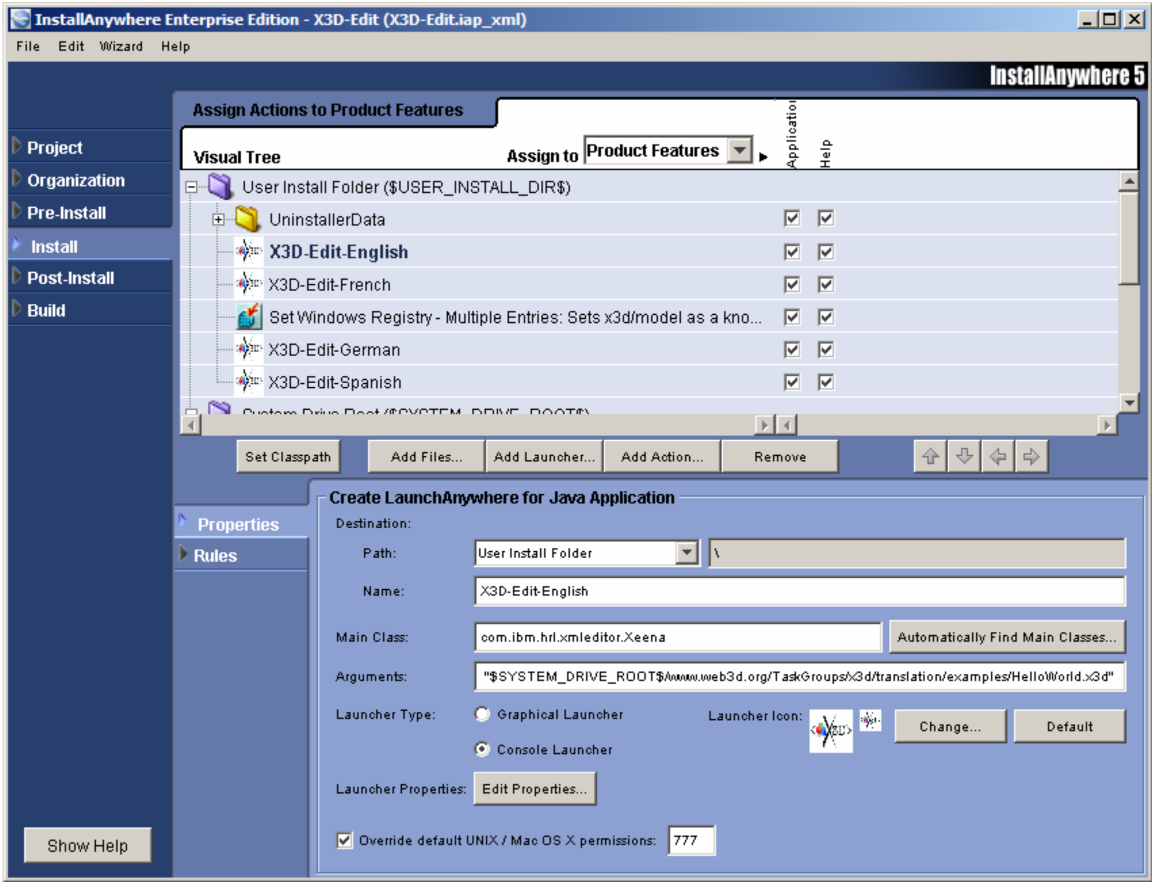


Figure 111. Depicts the configuration of Java Main Class arguments and default Mac OS X security permissions for the X3D-Edit InstallAnywhere project.

The next step on the installation configuration panel is to select the classpath settings to be used for our application deployment. These may be done in any combination of automatic or manual configuration and are explicitly annotated in the installation panel as to what is considered to be in the classpath of the deployment application (Figure 112).

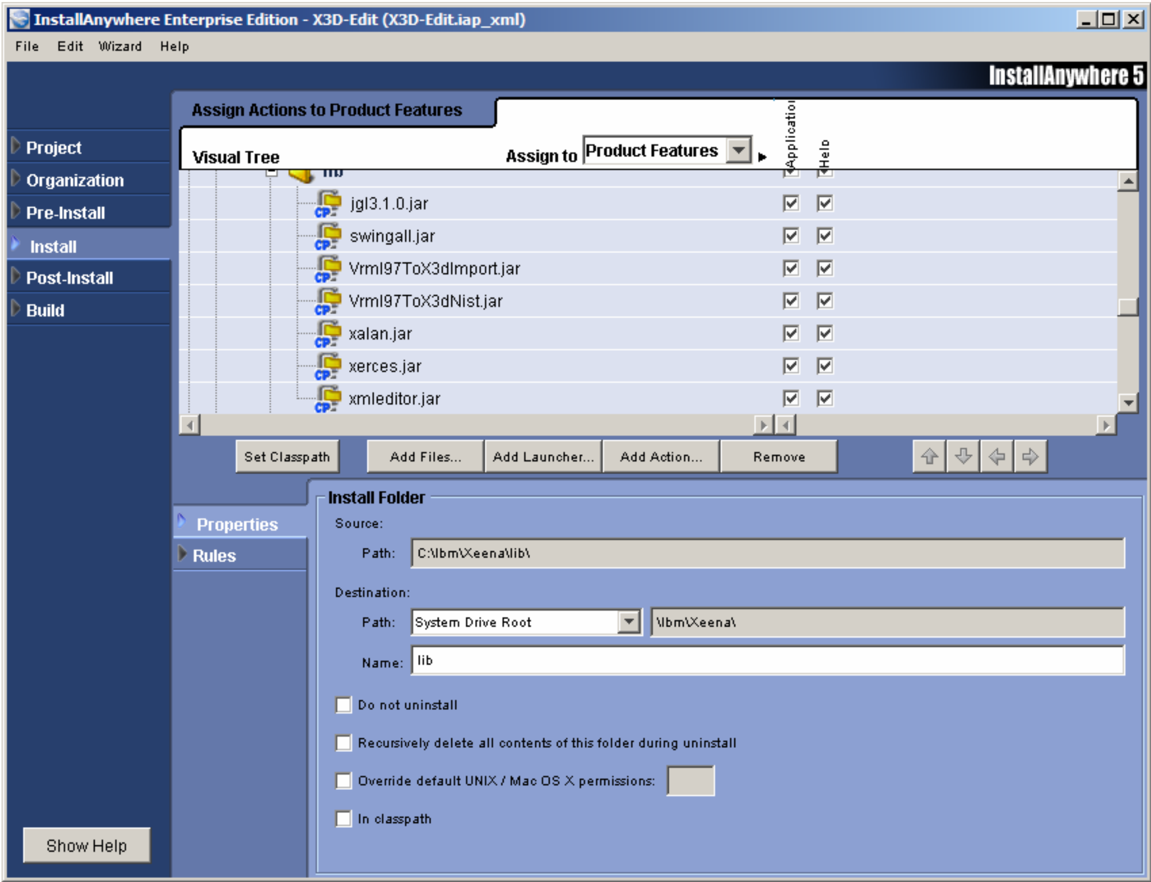


Figure 112. Depicts configuration of classpath for the X3D-Edit installation application in InstallAnywhere.

The next item we configure are the post-installation actions for the installation application to follow. Depicted in Figure 113, one can add additional web pages to be displayed to the end-user after the application has been installed but prior to completion of the installation steps. In this case, the X3D help.html is displayed to allow the user to select additional tools or resources to download prior to completing the final installation step.

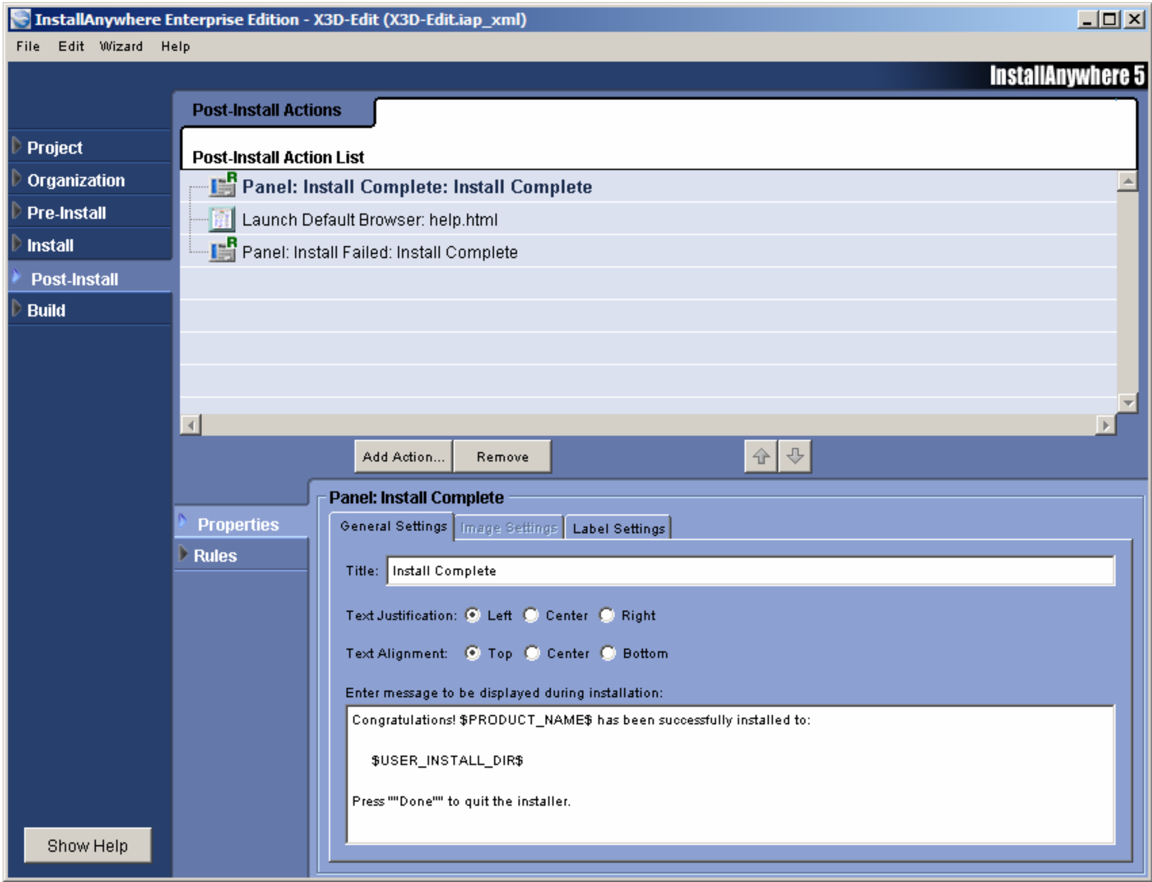


Figure 113. Depicts the Post-Installation configuration screen for InstallAnywhere.

The final step before proceeding to create and test a build for our application is to select the operating system and Java Virtual Machine (jvm) version(s) for each that will be deployed as options (Figure 114). To add additional versions, one can download what is referred to as a Virtual Machine (VM) Pack [ZEROG 2002] from http://www.zerog.com (accessed January 2003). Once downloaded and placed in the applicable directory and restarting the Install Anywhere application one may then use the new VM that has been downloaded. In order to add an extended virtual machine or a version that ZeroG has not included in their web-resources yet, one may simply create a zip archive of the desired Java Runtime directory, then include with a specified jvm manifest file, and then create a new archive with a .vm vice .zip extension. The new VM pack can be considered available for use after placing it in the applicable InstallAnywhere resource directory. This capability was utilized for the creation of a VM pack that included Java3D and Open GL for Java for the AT/FP Scenario Generation

Application Installer (Figure 115) with the specific process for creating and downloading new VM packs outlined in [ZEROG 2002]

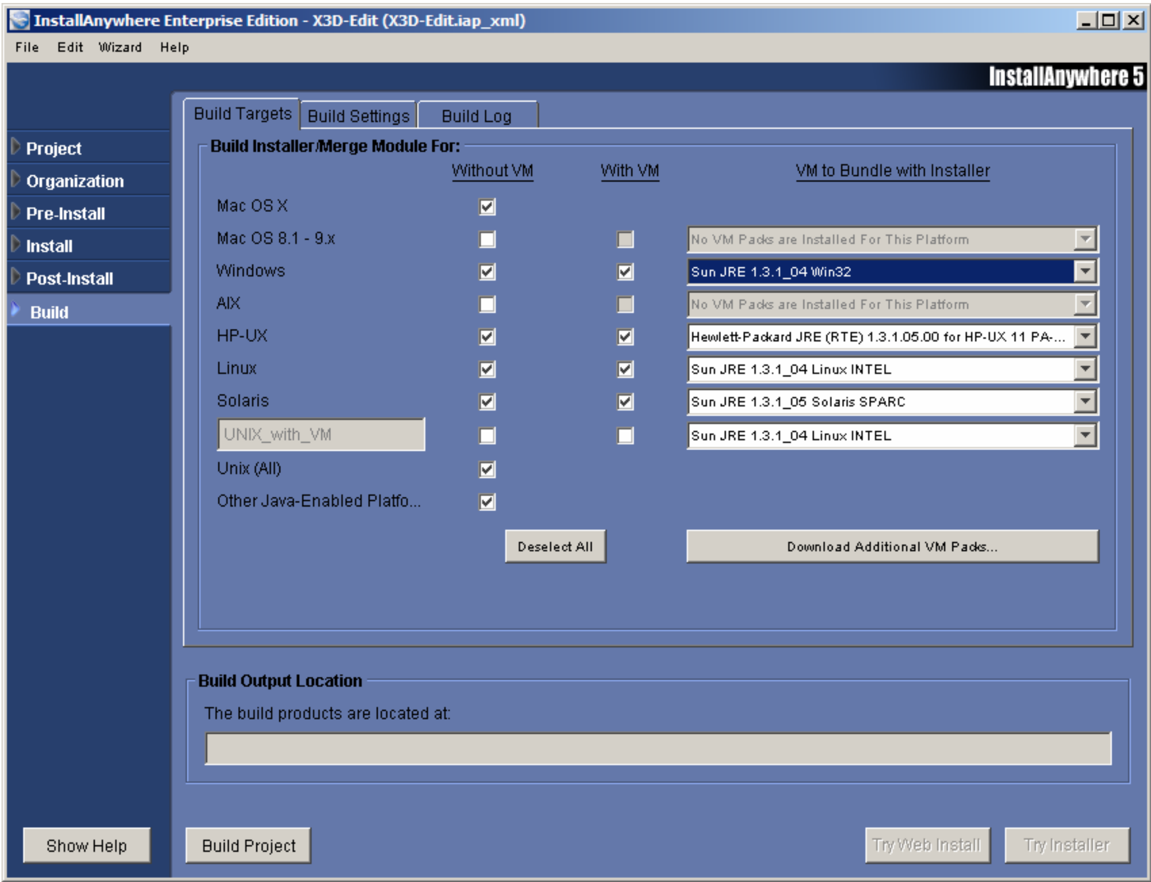


Figure 114. Depicts the Build Configuration panel in InstallAnywhere.

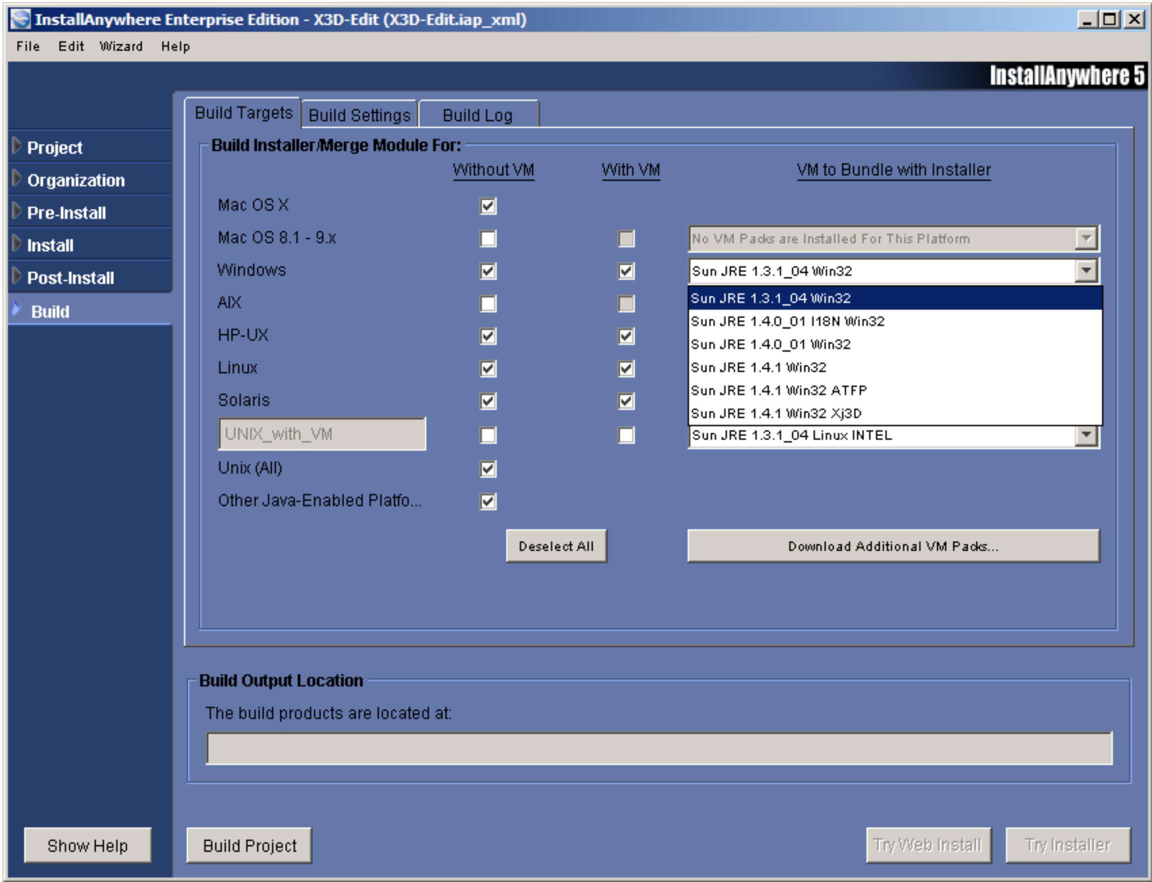


Figure 115. Depicts the selection of a VM from currently available ones on the development computer.

The final step is to build, run, and test the newly created installation applet. One can choose to create a web-enabled deployment applet, a compact disk (CD) deployment application, or both (Figure 116). The generated applet installation web-page was found to work satisfactorily in all available web browsers from Netscape versions 4.7x through 7.x, Internet Explorer versions 5.x-6.x, Mozilla, and the MSN 8 Explorer. Operating systems tested were Linux Red Hat 7.3, 8.1, Free BSD Unix, Solaris /Intel, Max OS X, Windows 95, 98, Me, 2000, and XP with success obtained for all after the aforementioned Windows inconsistencies were removed.

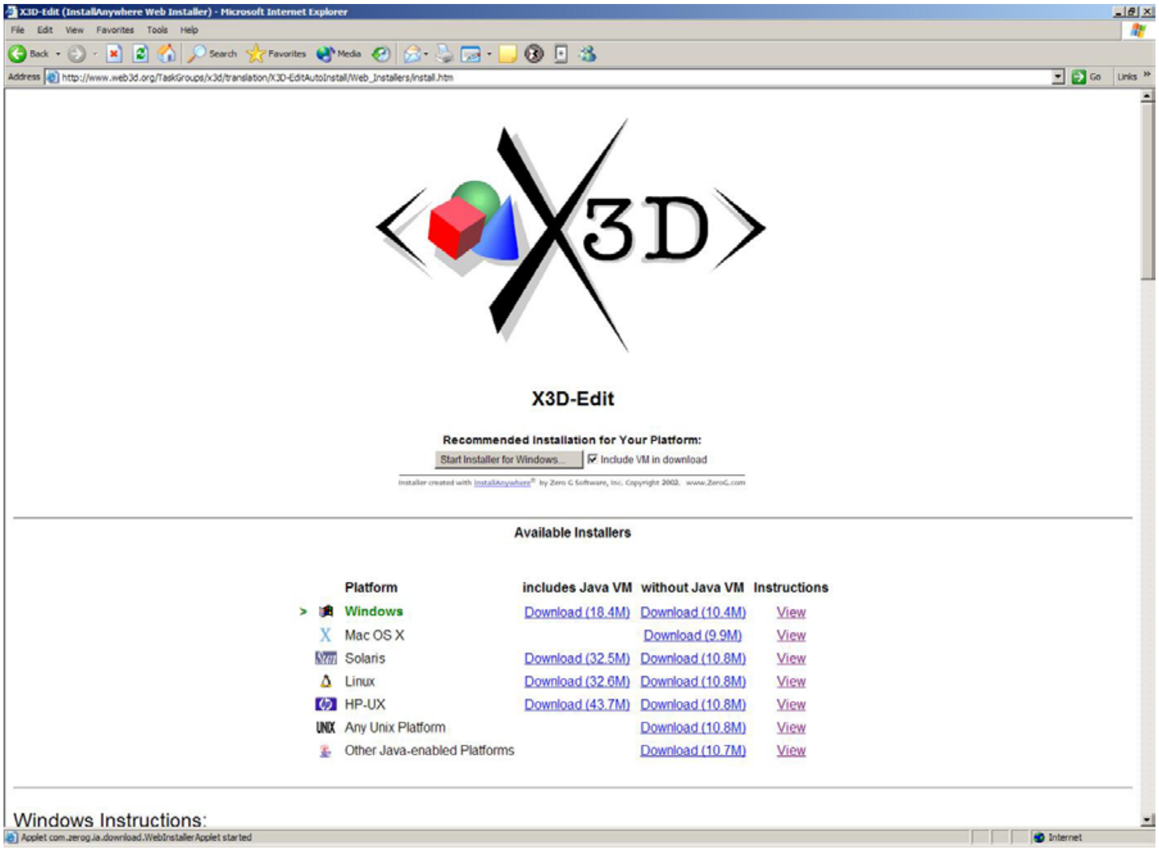


Figure 116. Depicts the X3D-Edit Installation Applet displayed in Netscape 7.01 running on a Windows XP Operating System.

D. INSTALL ANYWHERE AVAILABILITY

The commercially available Enterprise Edition of the Install Anywhere deployment solution was utilized for this appendix. The 1-seat NPS educational liscense cost approximately $2500 to include support and upgrades. Also available for no charge from ZeroG is the InstallAnywhereNow! Version of the application. This option is freely available with legal deployment applications allowed to be developed for a period of 30 days without risk of legal consequences. The Now! Version generally does not have as many features nor support options as the Enterprise Edition, but has and does serve as an alternative ‘light’ deployment option for those that might not have the monetary resources to acquire a commercial deployment option. [DICKIE 2002]

E. SUMMARY

In summary, the reader has been presented with an overview of the necessary steps to utilize the Install Anywhere Java Application Installer creation tool in context of creation of the open source X3D-Edit scene graph modeling tool maintained by the Web3D Consortium at http://www.web3d.org (accessed February 2003). Additional information on X3D may be obtained from Dr. Don Brutzman (brutzman@nps.navy.mil), http://web.nps.navy.mil/~brutzman/ (accessed 17 March 2003), and additional information on Install Anywhere from the ZeroG website at http://www.zerog.com (accessed February 2003).

APPENDIX M. LEVERAGING JAVA 2D FOR BASIC COLLISION DETECTION

A. INTRODUCTION

This appendix reviews how to utilize a subset of the Java2D API in order to implement basic collision detection and response using a harbor-based virtual environment as the driving application.

B. REPRESENTING GEOMETRY OFFSCREEN

The java.awt and java.awt.geom packages provide a standard set of 2D geometry classes that can be used to implement a decent level of fidelity for entity to entity and entity to terrain collision detection within a networked virtual environment. Although the Java3D API provides greater functionality and inherent knowledge of the third dimension, as developers we can not assume as widespread deployment of the 3D API, but can for the basic Java packages. As a result, the basic geometry classes provided in java.awt and java.awt.geom were investigated and leveraged to a satisfactory level of detail in order to effect basic collision detection and response for the small boat harbor environment utilized in this thesis. The reader is also referred to [Lamonde 2001] for additional information on the usage of the basic Java API in context of overall game or simulation design.

The first problem faced with implementation of collision detection in a water-borne environment without benefit of a game engine to optimize this task for us is to be able to adequately represent the water boundary with terrain or man-made structures such as piers. The decision was made to use the java.awt.Polygon class for this representation. The Polygon class is defined to be an encapsulation of a closed 2D region in our given coordinate space. [SUN 2003] We can have an arbitrary number of line segments, each being considered a side of our polygon. The internal representation of the polygon is made of a listing of (x,y) coordinates defining vertices of the polygon. Each successive pair of points define endpoints of a line segment that is a side of the polygon. The first and final points are automatically joined. [SUN 2003] We have available to us a constructor with the following signature with which to create a Polygon class instance:

Polygon(int [] xpoints, int [] ypoints, int npoints). The method takes an array of ints for the x-coordinates of the vertices, a corresponding array of y-coordinates for the vertices, and a third int value representing the total number of vertex points for the Polygon instance. One drawback to the Polygon class is that we do not have floating point or double values exposed to us for defining vertex point locations. This means that we have an error of up to 1 meter in length in the off-screen geometry definition which results in having to force a tighter error tolerance and resulting requirement of a smaller scenario timestep when less than one meter of accuracy is important in detecting collision with the land. We saw though, that in general for ships, they would run aground in the real world before this point in most cases so this constraint did not play an important factor in this work. But, looking at the signature of the class, the next problem was in how to develop the XZ plane vertex listing to represent the harbor environments. Two solutions were developed with future work identified for automating the process in some fashion. First, if a port is hand crafted from available information such as nautical charts, imagery, and design plans, then we have inherent knowledge of the off-screen coordinates and can plug in easily. For more complicated, or tool-generated environments, we found that we could accurately if tediously generate the coordinate listing by using Multigen Creator in vertex selection mode and select each vertex of interest to get the XZ plane coordinates for creating the off-screen Polygon object (Figure 117).

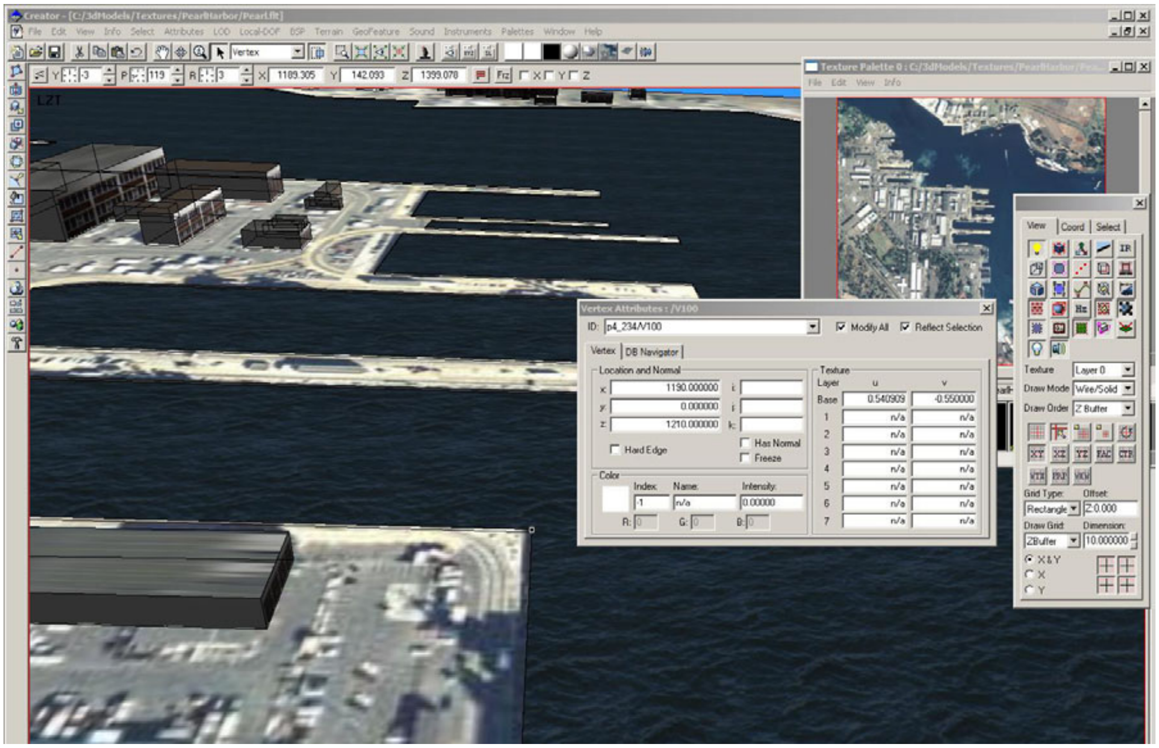


Figure 117. Depicts Multigen Creator in Vertex selection mode being used for creating the off-screen representation of the Pearl Harbor scene.

Once an off-screen polygon was created, we then have several methods that can be invoked by entities to determine if they have or are going to collide with the land (Table 3).

boolean	**contains**(double x, double y) Determines if the specified coordinates are inside this Polygon.
boolean	**contains**(double x, double y, double w, double h) Tests if the interior of this Polygon entirely contains the specified set of rectangular coordinates.
boolean	**contains**(int x, int y) Determines whether the specified coordinates are inside this Polygon.
boolean	**contains**(Point p) Determines whether the specified Point is inside this Polygon.
boolean	**contains**(Point2D p) Tests if a specified Point2D is inside the boundary of this Polygon.
boolean	**contains**(Rectangle2D r) Tests if the interior of this Polygon entirely contains the specified Rectangle2D

Table 3. Depicts a subset of the available methods for the Polygon class. From [SUN 2003]

Therefore, we can either represent a water-borne entity off screen with a java.awt.Recangle2D class object or its centroid with a java.awt.geom.Point2D object and make calls on the off screen Polygon object to see if we are still in the water (true) or not (false). Additionally, there is an intersect method that can be invoked with similar parameters available for usage. We found that this proved to be satisfactory for timesteps from 33 ms to 1 second in scenarios of up to twelve entities. Also, since all Java 2D geometry definitions of concern implement a common java.awt.Shape interface, the same methods for checking containment, distance, and intersection are exposed with all shape types, so the same methodology can be used to define simple bounding boxes to more complex off-screen representations of entities for implementation of entity to entity collision detection and response.

The next problem encountered was implementation of a more realistic collision response as a result of the collision with the land. For this, one has to be able to get a handle to the line segment that is being intersected. To this end, we can invoke the getPathIterator method illustrated in Table 4 in order to get an iterator to trace the boundary of the geometry to do an intersection test with our entity geometry.

PathIterator	**getPathIterator**(AffineTransform at) Returns an iterator object that iterates along the boundary of this `Polygon` and provides access to the geometry of the outline of this `Polygon`.

Table 4. Depicts the getPathIterator method for the Polygon class from [SUN 2003].

Once we have the applicable segment that we are colliding with, we can then calculate the surface normal to this segment and carry out any rigid body physics responses that we desire.

C. SUMMARY

In summary, the reader has been exposed to usage of the java.awt.Polygon class and java.awt.geom package for use in implementation of basic client-side java implementation of collision detection.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX N. LEVERAGING XSLT FOR X3D GRAPHICS DEPLOYMENT FOR HANDHELD DEVICES

A. INTRODUCTION

This appendix takes a short look at the portability of X3D graphics scenes combined with the use of XSLT for deployment to currently available Windows CE driven hand-held devices and notes both lessons learned as well as identifying areas for further research.

B. REAL-TIME 3D ON HAND HELD DEVICES

At SIGGRAPH 2000 in New Orleans, Louisiana, Parallel Graphics demonstrated the first web-based 3D rendering client for the Pocket PC/Windows CE operating systems with a public release following within a year called Pocket Cortona. [PARALLELGRAPHICS 2003] High end hand-held devices that could support reasonable performance for 3D graphics ranged in price from $600 to over $1000 until late 2002 when Dell began to sell the Dell Axim X-4 and X-5 handheld devices for prices approximately ranging from $150 to $300 before the addition of peripherals. Even more enabling than the reduction in price, however, was the increase in capability. The Dell Axim X-5, which was used for this study, was being sold in early 2003 with capabilities given in Table 5.

Processor	Intel XScale processor at 400MHx
Memory (RAM)	64 MB SDRAM
(ROM)	48 MB Intel StrataFlash ROM
Operating System	Microsoft Pocket PC 2002 Premium
Display Type	TFT Color 16-bit, touch sensitive, transflective display
Display Size	3.5 inches
Display Resolution	240 x 320 Pixels at 65,536 colors
Compact Flash	One Type II CompactFlash card slot
SecureDigital	One secure Digital card slot
Infrared port	Standard v1.3 (115 kbps)
Audio	Stereo headphone connector
Physical Dimensions	Length: 128 mm (5.04 inches) Width: 81.55 mm (3.21 inches) Height: 18 mm (.71 inches) Weight: 196 g (0.43 lb)
Audio Controller	AC-9 Codec chip; WM 9704 sound chip
Stereo Conversion	16-bit stereo; 8.0, 11.025, 22.05, and 44.1 KHz sample rates
Record	Full duplex record and playback
Microphone/speaker	Integrated
Headphone	Stereo connector.

Table 5. Depicts a subset of the hardware configuration of the Dell Axim X-5 from [DELL 2002]

Combined with the usage of an 802.11b wireless communications card, it was decided to investigate more fully the capabilities and limitations of the Axim X-5 with the Pocket Cortona VRML 97 client.

The first task to conduct was to determine any limiting factors on the VRML 97 specification implementation that Parallel Graphics took in the Pocket Cortona implementation. From, http://www.ParallelGraphics.com/products/cortonace/notes (accessed February 2003), the following items were not implemented in Pocket Cortona and no plans or dates were provided for possible future implementation: Java support in

the VRML Script node, External Authoring Interface (EAI) support in web pages, Movie Texture node implementation, alpha channel support in png image files, and no guaranteed support for ParallelGraphics extension nodes such as the Advanced Appearance node which contains BumpMapping and Multi-texturing capability, NURBS, etc. Although this was advertised, some support for the Collision Detection and Key-Sensor extension implementations was found. The only hardware limitations referenced were a minimum of 8 MB of RAM with 32 MB the recommended amount. General references were made to the support of Active-X controls within web-pages in Pocket Explorer, but no further information on how this might be leveraged for immediate usage was easily found.

Next, since we did not have the benefit of using Java in the Script node nor the EAI for implementing a direct port of this thesis work, we decided to implement an XSLT that transformed the native AT/FP XML scenario instance files to a static X3D scene representing the harbor and entity starting positions, which was then stylesheeted to VRML97 utilizing the X3DToVRML97.xslt provided by the Web3D Consortium as part of the X3D-Edit installation. Following this, we tried a simple rendering of the Pearl Harbor scene (Figure 118) with no ships present in the Pocket Cortona client.

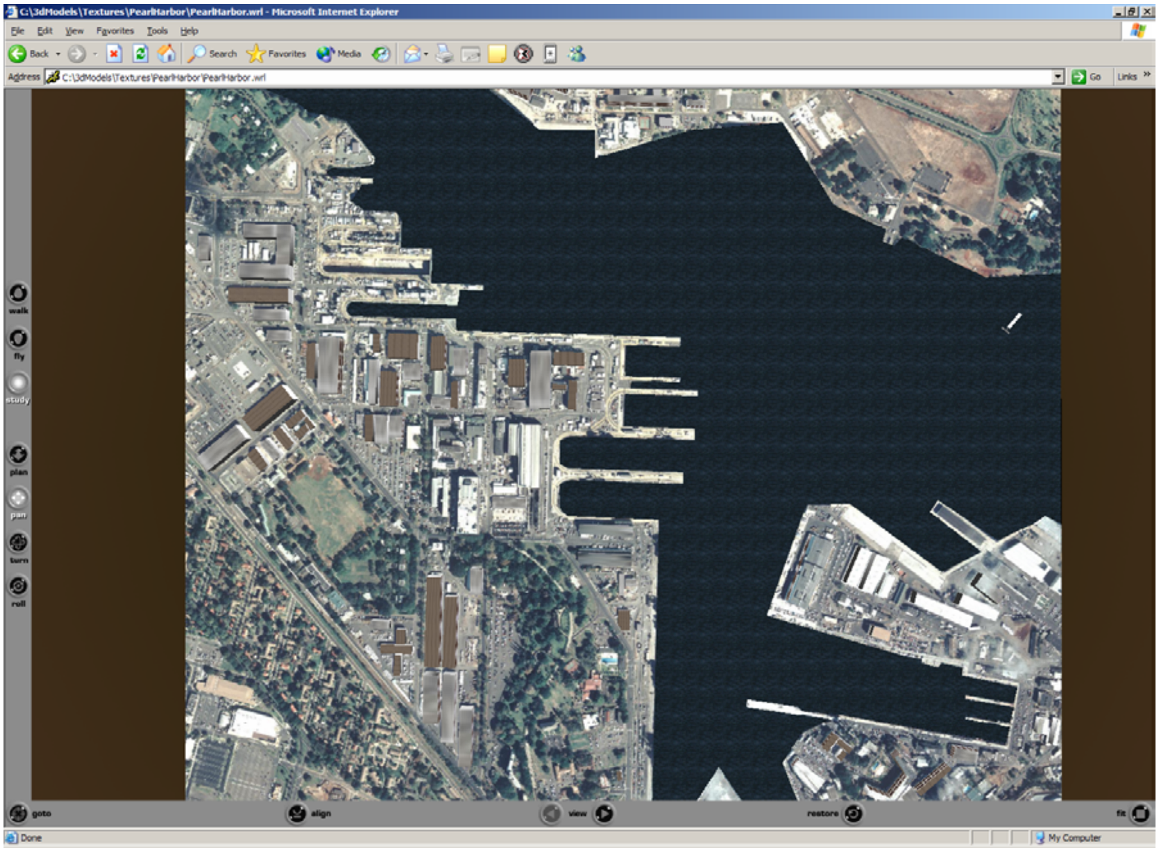


Figure 118. Depicts the PearlHarbor.wrl scene rendered in the Parallel Graphics Cortona VRML97 plugin for the PC.

The only problem experienced for this scene, was that the main texture size of 1.3 MB was too large for the client to display in a reasonable time frame for the scene. The Polygon count was approximately 7000 polygons with 300 kb of other textures utilized for the buildings and the water. So, the next step was to scale down the primary texture size by sixty percent. When this was done, the scene was able to be rendered at display rates averaging approximately 20 frames per second (fps) with this rate dipping as one navigated the scene via the stylus in examine mode. When pausing or selecting a predefined viewpoint, the display rate would generally start at around 20 fps in most cases. Of note, the Pocket Cortona implementation also supports interactivity to be programmatically added through the usage of EcmaScript in the VRML Script node, and the use of Routes amongst nodes which was used to explicitly bind the navigation info stack based upon the active viewpoint in the scene. (Figure 119)

Figure 119. Depicts the Arizona Memorial view in Pocket Cortona version 1.5 on the Dell Axim-5.

Next we investigated the capabilities for rendering the statically created scenarios with ships and geometry referred to above. The result was satisfactory, with the primary annoyance being a longer-than-expected time for file parsing and loading (Figure 120 below).

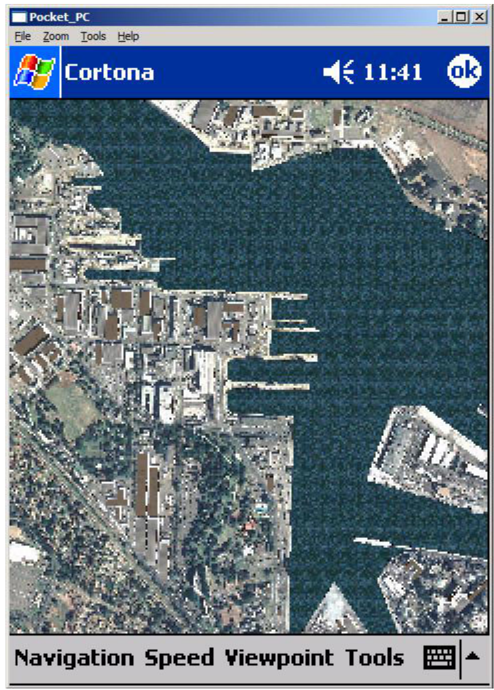


Figure 120. Depicts the PearlHarbor.wrl scene from Figure 118 displayed in Pocket Cortona on the Dell Axim Handheld device.

It was determined that although we could not provide real time simulation for viewing to the user without a means to open network sockets or otherwise communicate with our scenes to change position information on the fly, we could enable the end user to examine a harbor or port of interest in real-time 3D on the hand held devices which they could not have been able to do a few years ago. Combined with the Wireless networking capability, we could further leverage this capability by providing various hyperlinks to centralized data stores about the ports such as Port Security bulletins, tidal information, and so on.

C. LEVERAGING XSLT

Besides simply utilizing XSLT to style X3D geometry nodes from our scenario application files, we sought to identify any textures and or geometries that needed alternative 'light' versions to be created and referenced for hand held usage. The Pearl Harbor texture and large scale Aden, Yemen scenes were the primary two items that had to have alternative versions created in order to be useful on the hand held. [SUNWEB 2003] provides documentation on usage of the Apache Web Server to detect the

Operating System of the client-program navigating to our web site in order to deliver specifically styled content. As a result, if we were deploying this thesis in an Enterprise type of fashion which would have centrally dedicated servers, we could leverage the XSLT we've produced in this manner and further hide the fact that we are doing any extra work in order to deliver content for the Pocket PC.

D. OTHER MILITARY USES

Besides the usage leveraged for this thesis work, other potential military uses that Web3D technologies combined with the current hand held technologies are:

1 – Data Visualization – Modification and collaboration on CAD 3D drawings and design changes. [PARALLELGRAPHICS 2003]

2 – Field Maintenance – Providing in-the-field instruction and resources for conducting equipment maintenance and repair. [PARALLELGRAPHICS 2003]

3—Location Based Navigation [PARALLELGRAPHICS 2003]

4—Tactical Warfare Planning – Providing a 3D visualization of the battlespace while planning operational employment of forces with the integration of other planning tools.

E. FUTURE RESEARCH

The hardware appears to have reached a useful enough state that further research into alternative navigation and viewing paradigms focused on optimizing the end user's experience should be conducted soon. Combined with any results garnered from this area, this technology is ready to use now.

Conducting research into the use of the hand-held device as a controller for web-enabled 3D simulation systems also appears to be an interesting area to continue research outlined in [Watsen et, al. 1999].

F. SUMMARY

The reader has been exposed to the use of real time 3D graphics in ParallelGraphics Pocket Cortona VRML renderer on the DELL Axim X-5 hand-held, use of XSLT to create 'Pocket Friendly' versions of X3D and Vrml97 content, and some areas for future work. Use of handheld 3D graphics can aid in operator recognition of

unfamiliar landmarks from a sea-level perspective. Addition of existing GPS capabilities may provide further significant assists for AT/FP operators.

APPENDIX O. APPLICATION DISTRIBUTION AND SOURCE CODE ACCESS

All application source code, examples, and binary distribution are available via password protected access at: http://terra.cs.nps.navy.mil/SavageProjects/cd2/SavageProjects/harney/Harney.html (accessed March 2003) . Additionally, access to source code and distribution may be obtained from:

Dr. Don Brutzman: Brutzman@nps.navy.mil;

Research Professor John Hiles; jhiles@mindspring.com; or

Research Associate Curt Blais; clblais@nps.navy.mil.

Unrestricted access to versions of various components of the various 3D models developed during the conduct of this thesis are available as part of the SAVAGE 3D model distribution online at: http://web.nps.navy.mil/~brutzman/Savage/toc.html (accessed February 2003) and as part of the NPSNET-V distribution online at http://www.sourceforge.net/projects/npsnetv . (accessed March 2003)

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[Alexander 1977] Alexander, Christopher, *A Pattern Language: Towns, Buildings, Construction*, Oxford University Press, 1977.

[Alexander 1979] Alexander, Christopher, *The Timeless Way of Building,* Oxford University Press, 1979.

[Blais 2002a] Blais, C.L., Brutzman, D., Harney, J.W., & Weekley, J. Emerging Web-Based 3D Graphics for Education and Experimentation. In Proceedings, 2002 Interservice/Industry Training, Simulation, and Education Conference (Orlando, Florida, December 02-05). Available at http://www.movesinstitute.org/Publications/191_0816.pdf (accessed March 2003)

[Blais 2002b] Blais, C.L., Brutzman, D., Harney, J.W., & Weekley, J. (2002b). Web-based 3D reconstruction of scenarios for limited objective experiments. In Proceedings of the 2002 Summer Computer Simulation Conference, San Diego, 17-19 July. Available at http://www.movesinstitute.org/Publications/S192_Blais.pdf (accessed March 2003)

[Blais 2002c]Blais, C.L., Brutzman, D., Harney, J.W. & Hiles, J. Analyzing anti-terrorist tactical effectiveness for force protection using X3D graphics and agent based simulation. Presented at 70th MORS Symposium, Military Operations Research Society Conference. (Ft. Leavenworth, Kansas, June 16-18).

[Bourg 2002] Bourg, David M., *Physics for Game Developers*, Oreilly & Associates, Inc., Sebastopol, California, 2002.

[Brutzman 1998] Brutzman, D. (1998, June). The virtual reality modeling language and Java. Communications of the ACM, 41:6, 57-64.

[Burke 2001] Burke, Eric M, *Java and XSLT*, Oreilly& Associates, 2003, Sebastopol, California.

[CCOI 2001] USS Cole Court Of Inquiry, 2001. Previously online at: http://www.foia.navy.mil/usscole/index.html.
(Accessed January 2002) (Removed February 2002)

[Darken 2002] MV4202 Human Computer Interaction, Course Notes 2002, Rudolph Darken.

[DELL 2002] *Dell Axim X5 User's Guide*, 2002, Dell Computer Corporation, http://www.dell.com (accessed February 2003).

[Dickie 2002] Dickie, Alistair (2002), Modeling Robot Swarms Using Agent -Based Simulation, Master's Thesis, Naval Postgraduate School. Available at: http://www.movesinstitute.org/Theses/AlistairDickie.pdf (accessed March 2003)

[DTED 1996] Performance Specification, Digital Terrain Elevation Data (DTED), 19 April 1996, MIL-PRF—89020A, Defense Mapping Agency, Fairfax, Virginia, US.

[Eisenberg 2002] Eisenberg, J. David, *SVG Essentials*, 2002, O'Reilly & Associates, INC., Sebastopol, California.

[Fauconnier 2001] Fauconnier, Gilles, Turner, Mark, Conceptual Integration Networks, published in Cognitive Science, 22(2) 1998, 133-187, Cognitive Science Society, Inc. Current updated and expanded web version 20 February 2001, available at http://www.wam.umd.edu/~mturn/WWW/blending.html (accessed February 2003).

[Ferber 99] Ferber, J., *Multi-Agent System, An Introduction to Distributed Artificial Intelligence.* Addison-Wesley Publishers, 1999.

[Gomez 2000]Gomez, Miguel, "Integrating the Equations of Rigid Body Motion", from *Game Programming Gems*, Charles River Media, Inc, Rockland, Massachusettes, 2000.

[Grand 1998] Grand, Mark. *Patterns in Java, Volume 1.* John Wiley & Sons, Inc., New York, 1998.

[GUARDIAN 2002] "Al-Qaida Suspected in Tanker Explosion", Guardian Unlimited, 7 October 2002. (accessed March 2003) http://www.guardian.co.uk/alqaida/story/0%2C12469%2C805997%2C00.html.

[Hiles 2002] MV4015 Autonomous Agents, Course Notes 2002, John Hiles.

[Hiles 2003] Hiles, John, draft white paper, Integrated Asymmetric Goal Organization (IAGO): A Multi-agent Model of Conceptual Blending, 2003.

[Hunsberger 2001] Hunsberger, M. G. (2001, June). 3D Visualization of Tactical Communications for Planning and Operations Using Virtual Reality Modeling Language (VRML) and Extensible 3D (X3D). Monterey, California: Naval Postgraduate School.

[Hundt 2002] Spartan Scout, Advanced Technology Concept Demonstration, FY 2002 ACTD, September 2002.

[Hunter 2001] Hunter, et. al., David; Cagle, Kurt; Dix, Chris; Kovack, Roger; Pinnock, Jonathan, Rafter, Jeff; Beginning XML (2nd Edition), 2001, Wrox Press, Birmingham, UK.

[INTEL 2002] Press Release, 23 July 2002, "Intel Establishes Working Group To Create Standards For 3D On The Web", San Antonio, Texas. http://www.intel.com/pressroom/archive/releases/20020723corp.htm (accessed October 2002).

[JNLPSPEC 2001] Java Network Launching Protocol and API Specification, (JSR-56), 2001, version 1.0.1, Sun Microsystems.

[JP1 2002] Joint Publication 1-02, DoD Dictionary of Military and Associated Terms.

[JMOCN 2002] Naval War College Course Notes, Joint Military Operations, Segment 4.3 Combating Terrorism, 2002.

[JV2020 2000] Shelton, H., Joint Vision 2020, 2000

[Lamonde 2000] Lamonde, Andre, *Tricks of the Windows Game Programming Gurus, Fundamentals of 2D and 3D Game Programming,* Sams, Indianapolis, Indiana, 1999.

[Lamonde 2001] Lamonde, Andre, *Tricks of the Java Game Programming Gurus,* Sams, Indianapolis, Indiana, 2000.

[LINUX 2002] Goodman, Adam M., “As the Linux World Turns”, *Linux Magazine*, October 2002, www.linuxworld.com (accessed January 2003).

[Kay 2001] Kay, Michael, *XSLT Programmer’s Reference 2nd Edition,* Wrox Press, 2001, Birmingham, United Kingdom.

[Mangano 2003] Mangano, Sal, *XSLT Cookbook*, O’Reilly & Associates, 2003, Sebastopol, California.

[Mnif 2003] Mnif, Khaled, Using XML/HTTP to Store, Serve, and Annotate Tactical Scenarios for X3D Operational Visualization and Anit-Terrorist Training, 2003, Master’s Thesis, Naval Postgraduate School.

[Murray 2000] Murray, M. W., & Quigley, J. M. (2000, June). Automatically Generating a Distributed 3D Battlespace Using USMTF and XML-MTF AIR Tasking Order, Extensible Markup Language (XML) and Virtual Reality Markup Language (VRML). Naval Postgraduate School.

[Nicklaus 2001] Nicklaus, S. (2001). Scenario Authoring and Visualization for Advanced Graphical Environments. Master’s Thesis, Naval Postgraduate School. Available at: http://www.movesinstitute.org/Theses/ShaneNicklaus.pdf (accessed March 2003)

[Nielsen 2000] Nielsen, Jakob, *Designing Web Usability,* New Riders Publishing, Indianapolis, Indiana, 2000.

[Neushul 2003] Neushul, James (2003) Draft Master's Thesis, Naval Postgraduate School.

[Osborne 2002] Osborne, Brian (2002), An Agent-Based Architecture for Generating Interactive Stories, Dissertation, Naval Postgraduate School. Available at: http://www.movesinstitute.org/Theses/OsbornDissertation.pdf (accessed March 2003)

[OSI 2002] Open Source Initiative, Non-Profit Corporation, 2002, "Definition and Rationale", http://www.opensource.org (accessed September 2002).

[PARALLELGRAPHICS 2003] Pocket Cortona Release and User's Notes, created 2001, revised through 2003, online at http://www.parallelgraphics.com/products/cortonace/notes (accessed February 2003).

[SUN 2003] Sun Microsystems, Java 2 SDK Documentation, Version 1.4.1, 2003.

[SUNWEB 2003] The Java Web Services Tutorial, A beginners guide to developing Web services and Web applications on the Java Web Services Developers Pack, Sun Microsystems, 2002.

[USNA 2000] ES310 Introduction to Naval Weapons Engineering Course Notes, U.S. Naval Academy, 28 January 1998, http://www.fas.org/man/dod-101/navy/docs/es310/syllabus.html (accessed March 2003).

[USNI 2001] "Twenty Four Hours to Go" The USS McCampbell (DDG-85) prepares for her launch, U.S. Naval Institute, April 2001, Volume 127/4/1, 178, Photograph by Steven Footer, (accessed March 2003) http://www.usni.org/Proceedings/Articles01/PROphotocontest4.html .

[X3D 2002] Web3D Consortium Press Release, 23 July 2002, "Web3D Consortium Releases X3D Final Working Draft", San Antonio, Texas.

[X3DSPEC 2002] X3D Specification (Draft), http://www.web3d.org/specs/ (accessed September 2002).

[XMSF 2002] Brutzman, D., Morse, K., Pullen, M., Zyda, M., "XMSF 2002 Findings and Recommendations Report", Technical Challenges Workshop and Strategic Opportunities Symposium, 22 October 2002. Available at: http://www.movesinstitute.org/private/tmp/501/Temporary%20Items/Library/Apache2/htdocs/xmsf/XmsfWorkshopSymposiumReportOctober2002.pdf (accessed February 2003)

[XMSF 2002a] Extensible Modeling and Simulation Website, available at: http://www.movesinstitute.org/xmsf/xmsf.html (accessed March 2003)

[Watsen, et al. 1999] Watsen, K., Darken, R., Capps, M., "A Handheld Computer as an Interaction Device to a Virtual Environment", Proceedings of the International Projection Technologies Workshop, Stuttgart, Germany, May 10-11, 1999. Available at: http://www.movesinstitute.org/darken/publications/PalmPilot.pdf (accessed March 2003)

[Wellbrink 2003] Wellbrink, Joerg, Darken, Rudy (2003). Modeling Reduced Human Performance as Complex Adaptive System. Draft Paper submitted to: Behavior Representation in Modeling and Simulation Scottsdale, Arizona SISO.

[ZEROG 2002] Zero G Software, Install Anywhere 5 User Guide, 2002, San Francisco, California.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Fort Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Admiral Robert J. Natter
   Commander U.S. Fleet Forces Command,
   Commander U.S. Atlantic Fleet
   Norfolk, Virginia

4. Vice Admiral Mike Bucchi
   Commander U.S. Third Fleet
   San Diego, California

5. Vice Admiral Timothy W. LaFleur
   Commander U.S. Naval Surface Forces Pacific
   San Diego, California

6. Vice Admiral Cutler Dawson, Jr.
   Commander U.S. Second Fleet / NATO Striking Fleet Atlantic
   Norfolk, Virginia

7. Rear Admiral Terrance T. Etnyre
   Commander U.S. Naval Surface Forces Atlantic
   Norfolk, Virginia

8. Rear Admiral Ronald A. Route
   Commander, Naval Warfare Doctrine Command
   Newport, Rhode Island

9. Rear Admiral Henry G. Ulrich, III
   Director, Surface Warfare Division (OPNAV N76)
   Washington, D.C.

10. Ms. Celia Metz
    SPAWAR, Homeland Defense Office Chairman
    San Diego, California

11. Professor Michael J. Zyda,
Chair, Modeling, Virtual Environments and Simulation (MOVES)
Naval Postgraduate School
Monterey, California

12. Associate Professor Don Brutzman
Naval Postgraduate School
Monterey, California

13. Research Professor John Hiles
Naval Postgraduate School
Monterey, California

14. Research Associate Curt Blais
Naval Postgraduate School
Monterey, California

15. Research Associate Doug Horner
Naval Postgraduate School
Monterey, California

16. Research Associate Jeff Weekley
Naval Postgraduate School
Monterey, California

17. Professor Gordon Schacher
Naval Postgraduate School
Monterey, California

18. Professor Don MacGregor
Naval Postgraduate School
Monterey, California

19. Research Associate Andrezj Kapolka
Naval Postgraduate School
Monterey, California

20. Research Associate Barb Helfer
Naval Postgraduate School
Monterey, California

21. Jeff Kline, CAPT USN
Naval Postgraduate School
Monterey, California

22. James Harney, LT USN
    Naval Postgraduate School
    Monterey, California

23. Dr. Roy S. Harney, Jr.
    Nicholasville, Kentucky

24. Alan Hudson
    Yumetech Inc.
    Seattle, Washington

25. Justin Couch
    Yumetech Inc.
    Seattle, Washington

26. Rick Goldberg
    Aniviza Inc.
    Santa Clara, California

27. Ray Waters
    Naval Warfare Doctrine Command, AT/FP Warfare Innovation Team
    Newport, Rhode Island

28. Robert Gregory
    AT/FP School, EWTGLANT
    Chesapeake, Virginia